MANIPULATING OBJECTS USING COMPLIANT, UNACTUATED TAILS: MODELING AND PLANNING

A Dissertation

by

YOUNG-HO KIM

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Dylan A. Shell
Committee Members,	Ergun Akleman
	Ricardo Gutierrez-Osuna
	Dezhen Song
Head of Department,	Dilma Da Silva

May 2017

Major Subject: Computer Engineering

ABSTRACT

Ropes and rope-like objects (*e.g.,* chains, cords, lines, whips, or lassos) are comparatively cheap, simple, and useful in daily life. For a long time, humans have used such structures for manipulation tasks in a qualitatively different ways such as pulling, fastening, attaching, tying, knotting, and whipping. Nevertheless, these structures have received little attention in robotics. Because they are unactuated, such structures are regarded as difficult to model, plan and control. In this dissertation, we are interested in a mobile robot system using a flexible rope-like structure attached to its end akin to a 'tail'.

Our goal is to investigate how mobile robots can use compliant, unactuated structures for various manipulation tasks. Robots that use a tail to manipulate objects face challenges in modeling and planning of behaviors, dynamics, and combinatorial optimization. In this dissertation, we propose several methods to deal with the difficulties of modeling and planning. In addition, we solve variants of object manipulation problems wherein multiple classes of objects are to be transported by multiple cooperative robots using ropes.

Firstly, we examine motion primitives, where the primitives are designed to simplify modeling and planning issues. We explore several sets of motion primitive where each primitive contributes some aspect lacking in the others. These primitives are forward models of the system's behavior that predict the position and orientation of the object being manipulated within the workspace. Then, to solve manipulation problems, we design a planner that seeks a sequence of motion primitives by using a sampling-based motion planning approach coupled with a particle-based representation to treat error propagation of the motions. Our proposed planner is used to optimize motion sequences based on a specified preference over a set of objectives, such as execution time, navigation cost, or collision likelihood. The solutions deal with different preferences effectively, and we an-

alyze the complementary nature of dynamic and quasi-static motions, showing that there exist regimes where transitions among them are indeed desirable, as reflected in the plans produced.

Secondly, we explore a variety of interesting primitives that result in new approaches for object manipulation problems. We examine ways two robots can join the ends of their tails so that a pair of conjoined robots can encircle objects leading to the advantage of greater towing capacity if they work cooperatively. However, individual robots possess the advantage of allowing for greater concurrency if objects are distant from one another. We solve a new manipulation problem for the scenarios of moving a collection of objects to goal locations with multiple robots that may form conjoined pairs. To maximize efficiency, the robots balance working as a tightly-knit sub-team with individual operation. We develop heuristics that give satisfactory solutions in reasonable time. The results we report include data from physical robots executing plans produced by our planner, collecting objects both by individual action and by a coupled pair operation.

We expect that our research results will help to understand how a flexible compliant appendage when added to a robot can be useful for more than just agility. The proposed techniques using simple motion models for characterizing the complicated system dynamics can be used to robotics research: motion planning, minimalist manipulators, behavior-based control, and multi-robot coordination. In addition, we expect that the proposed methods can enhance the performance of various manipulation tasks, efficient search, adaptive sampling or coverage in unknown, unstructured environments.

ACKNOWLEDGEMENTS

Looking back over my doctoral period for the last six years, I would not have completed my long and lonely journey, a doctoral program, without the help of many people including my advisor, committee members, colleagues, friends, and family. I would like to express my gratitude to them in this section. I will try to mention all of them, but please note that it is likely I accidentally omitted some.

I would like to sincerely thank my academic father, Dr. Dylan A. Shell for his endless devoted help for the academic achievement and doctoral life's uncertainties over 6 years. His constant encouragement as being the best researcher has always been a positive influence on my whole doctoral period. He has helped me to find my own answers by asking insightful questions rather than giving answers. In particular, he supported me from the start of my doctorate period so that I only can focus on my research. Although many of the contributions presented in my doctoral research are derived from his mentorship, I believe that all the research training that he has provided will be a great asset to my next career as a researcher. I will be forever indebted to him.

I would like to thank my committee members, Dr. Ricardo Gutierrez-Osuna, Dr. Dezhen Song, and Dr. Ergun Akleman for their insightful comments and constructive criticisms to my research for better organizing my dissertation. Especially, during the qualifying and prelim exams, without their constant encouragement and support, I could not have completed my degree program.

I would like to thank my fellow students in the Distributed AI Robotics Lab for their support, helpful discussions, and friendship during my years in the doctoral program: Jung-Hwan Kim, Changjoo Nam, Yong Song, Lantao Liu, Ben Fine, Sasin Janpuangtong, Yulin Zhang, Reza Oftadeh, Eric Cochrane, Taahir Ahmed, Asish Ghoshal, Plamen Ivanov, Reza

Hosseini Teshnizi, Shawn Kristek, Tanushree Mitra, Jing Zhou, and Rui Liu. I also want to thank Sang-Wook Lee, Joseph Lee, Byung-Jun Yoon, Byung-Hak Kim, Yan Lu, Han Ul Yoon, Sungtae Shin, and Woohyun Ko for helpful discussions and friendship during my doctoral period. Many friends from Korean Church A&M gave me strength to keep my spiritual life from getting tired and also made my graduate life enjoyable. I can not comment on everyone, but I would like to take this opportunity to thank you. I am also very grateful for the help provided by the administrative staff of the Department of Computer Science and Engineering.

Last but not least, I would have never been able to accomplish anything in my life without the devoted endless support of my family. I thank my parents' love, prayers, and encouragement. I thank my sisters Young-Mi and Young-In for having always believed in me to pursue my dreams. I thank my mother-in-law Haeran and sister-in-law Yoon for endless love, support, and prayer. Most importantly, I would like to thank my most precious, pretty, and wise wife, Jung. She always believed and encouraged me, and whenever I was dejected, she has always been with me. Lastly, to my lovely kids Jion and Jian, I always got strength from you two. Without your sweet strong hug, I could not endure my long doctoral period. I thank God for everything that I have been given during my doctoral period.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor Dylan A. Shell [advisor], Professor Ricardo Gutierrez-Osuna, and Professor Dezhen Song of the Department of Computer Science and Engineering and Professor Ergun Akleman of the Department of Visualization.

All works conducted for the dissertation were completed by the student independently.

Funding Sources

Graduate study was supported by the National Science Foundation (NSF) through grants IIS-1302393 and IIS-1453652.

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

TABLE Page

1. INTRODUCTION

1.1 Motivation: Manipulating Objects Using Tools

Object manipulation has been a major subarea of robotics research for several decades. Most people are familiar with dexterous arm manipulators, which can move a specific object from one location to another [1, 4] (shown in Figure 1.1(a)). These robots have already been applied to industrial settings, for example in factories, where they have been used to improve manufacturing productivity by performing many simple tasks in a repeatable fashion. In recent robotics research, mobile robots have been carrying out difficult tasks (*e.g.,* rearranging objects in warehouses, drilling holes, or unscrewing a cap) using various tools, which have been usually employed by humans. Figure 1.1(b) shows a mobile humanoid robot using an electric handdrill to make a hole in a wall [2].

(a) A robotic arm manipulator moves varios objects to goal locations in DARPA Robotic Manipulation 2012 [1].

(b) Robots use many kinds of tools to complete tasks in DARPA Robotics Challenge 2015 [2].

(c) A baboon uses a tool for capturing termites [3].

Figure 1.1: Manipulating objects using tools: (a) A dexterous arm manipulator moves a specific object from one location to another location. (b) Robot using a tool to drill into a wall. (c) Some animals use tools for prey capture, demonstrating greater intelligence.

The use of elaborate tools has been regarded as mostly the exclusive capability of humans. Only limited species of animals use tools for prey capture (Figure 1.1(c)) to improve their given capabilities [3]. Our work focuses on how robots can use tools. More specifically, our goal is to investigate how a mobile robot may use a flexible rope-like structure attached as a 'tail' that it can use as a tool.

From the ancient Egyptian civilization to modern times, humans have used ropes and rope-like objects (*e.g.,* chains, cords, lines, whips, or lassos) for manipulation in a variety ways such as pulling, fastening, attaching, carrying, lifting, and climbing. For example, people working a kelly drive drilling rig (shown in Figure 1.2(a)) explore for oil by making a connection to continue drilling downward. In this case, the flexible structure (a chain) enables the worker to exploit constrictional, tensional, and frictional forces to restrain the object being manipulated, sometimes also helping keep the object under control by gripping it statically [5]. During World War II, people developed a mine flail, which consists of a number of heavy chains, dynamically spinning and pounding the ground to detonate a buried mine [6] (shown in Figure 1.2(b)). On the surface of water, floating flexible structures are used to drag water hyacinth by two tethered boats [7] (shown in Figure 1.2(c)). In martial-arts cinema portrayals, whipping actions are most commonly used for reaching and attacking enemies (*e.g.,* Indiana Jones's famous bull whip [8], cowboys with their lassos [9], and Spiderman [10] as well). Those whip-like actions shown in Figure 1.2(d), in contrast to winding and tying actions, are produced with flexible cords by exploiting the dynamics of their continuous, compliant structures.

However, despite offering versatility, such flexible, unactuated structures have received little attention in robotics research. We think that the reason is that these passive structures are unactuated, thus difficult to control. Nevertheless, they are comparatively simple and cheap, and are easy to find (*e.g.,* cords, belts, and even shoe laces). In addition, these flexible structures are useful as tools with versatility in our daily lives. Also, these structures

minimize dependence on the shape of objects. Consequently, if we can use these flexible structures as tools for robots, then one may avoid the cost of a special gripper.

1.2 Challenges: Statement of the Problem

Next, we take a closer look at why rope-like structures as tools are rarely used in real robot systems. We hypothesize that a robot with a compliant passive structure attached as a tail are seldom used for manipulation since modeling (below (1) to (3)) and planning (below (4) to (5)) are difficult:

(1) **System Modeling:** The interplay of the object, the robot, and its tail involves mutual influences that require non-trivial physics in order to describe their interactions. This complicates the modeling of joint states (the object, robot, tail trio) and makes their full description fairly daunting.

(2) **Dynamics:** An underactuated tail imposes severe limits on the scope and precision with which the manipulator's configuration can be controlled.

(3) **State Estimation:** The object's state transitions are governed by the tail, which imposes a level of indirection —the tail itself being mediated by the robot's motion— that means the transitions are non-deterministic.

(4) **Planning and Decision-making:** Considering all possible motions with non-deterministic state transitions, sequencing these motions to accomplish given tasks is complicated because there exist various interrelated objectives such as navigation cost, execution time, or reliability.

(5). **Coordination:** When there are multiple objects to be manipulated, multiple robots using rope-like structures can be used in can be used in variety ways (*cf.* Donald et al. [11], Bhattacharya et al. [12]). Many interesting combinations of robots using rope-like structures can be designed. For one typical example, imagine two robots can join the ends of their rope-like structures, then the conjoined robots can encircle multiple objects. To maximize efficiency, one might examine all combinations of multiple robots and objects, which is a computationally challenging task.

1.3 Research Objective

In this dissertation, our goal is to investigate a novel robot system with flexible tail for manipulating objects in various ways. Questions we plan to answer include: (1) Which motions can manipulate objects with what properties? And then, (2) how do we model those motions? (3) How do we plan a path using those motions for specific manipulation tasks? (4) How does a group of robots with flexible structures form a team to solve multi-object collection tasks? (5) How do we assign tasks to a group of robots to optimize performance of the multi-object collection problems?

1.4 Overview of Research Approach

Firstly, we explore possible motions by mobile robots with a flexible structure. Then, we design motion primitives for manipulating objects, where the primitives simplify modeling and planning issues. We demonstrate several sets of diverse motion primitives where each contributes some aspect lacking in the others. These primitives are forward models of the systems' behavior that predict the object positioning and orientation within the workspace. Then, to solve the manipulation problem, we design a planner to seek a sequence of motion primitives by using a sampling-based motion planning approach coupled with a particle-based representation to treat error propagation of the motions. Our proposed planner optimizes motion sequences based on a specified preference over a set

of objectives, such as execution time, navigation cost, or collision likelihood. The results of such a planner deal with diverse preferences effectively, and we analyze the complementary nature of dynamic and quasi-static motions, showing that there exist regimes where transitions between the two are indeed desirable, as reflected in the plans produced.

Secondly, to deal with various object manipulation problems, we examine the manipulation problem of moving a collection of objects to multiple goal locations with multiple robots. To maximize efficiency, the robots must balance working as a tightly-knit sub-team versus operating individually. Pairs have the advantage of greater towing capacity, while individuals can fetch distant objects at the same time. We formulate the planning problem for efficiently collecting multiple objects and transporting them to goal locations. We do this within a general framework using logical formulas to express complex tasks. Since this problem is proved as NP-hard, we explore heuristics that give satisfactory solutions in reasonable time. The results we report include data from physical robots executing plans produced by our planner, collecting objects both by individual action and by operation of coupled pairs.

1.5 Contributions

The main contributions of this dissertation are:

- We make static, quasi-static, and dynamic models for mobile robots using flexible rope-like structures: a) We use dynamic motions shown to be efficient for some objectives (*e.g.,* object distance per action); b) Our planning algorithm allows robots to employ a diverse set of motions showing each motion has distinct complementary value to one another; c) Our physical experiments show a successful demonstration of manipulation therewith, the first to exploit the dynamics of a linear structure.

- We study a coordinated towing system where: a) All robots can be separated or conjoined; b) This is chosen automatically by the algorithm; c) This is dynamic: changes

during execution— sometimes working as individually or sometimes operating as a tightly-knit sub-team. We propose an efficient algorithm to reduce search space in various environmental settings. We show the first known physical demonstration of multiple robots solving manipulation problems in this way.

- We consider two problems together: multi-robot motion planning and logical specification of plan goals. We introduce a generalized framework using both in a multi-robot manipulation setting, the first of this sort.

1.6 Dissertation Organization

In the following chapters, we demonstrate a new approach to employ flexible rope-like structures as tools for various manipulation tasks. Section 2 details the current state of robots with tools, and manipulating objects in terms of modeling and planning systems. Section 3 introduces modeling and planning methods for moving a single object by the flexible rope-like structures.We propose an algorithm to use a diverse set of motion primitives where each primitive contributes unique aspect lacking in the other primitives. In Section 4, we formulate the planning problem for efficiently collecting multiple objects and transporting them to goal locations, and then we explore analysis of the proposed algorithms. The results we report include data from physical robots executing plans produced by our proposed planner, collecting objects both by individual action and by a coupled pair action. Finally, we conclude and present future works in Section 5.

(a) People run a kelly drive drilling rig exploring for oil by wrapping a chain around a pipe [5].

(b) Rapidly rotating a number of heavy chains is to detonate a buried mine [6].

(c) Water hyacinth is dragged by ropes [7].

(d) Whipping actions used for reaching and grapping objects [8].

Figure 1.2: Motivating examples of manipulation with ropes and rope-like structures: they are a class of useful tools possessing unrivaled versatility as used in wrapping, dragging, tying, knotting, whipping.

2. RELATED WORK

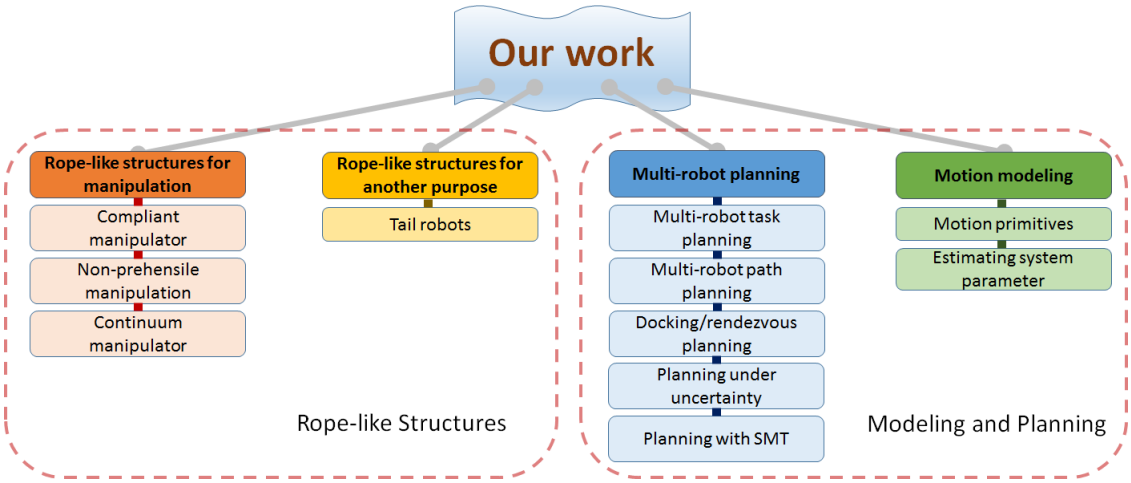


Figure 2.1: Overview of Related Work

Our research lies at the intersection of compliant manipulator, continuum manipulator, non-prehensile manipulator, motion primitive, active tails, and multi-robot motion planning. Broadly speaking, we can split them into two categories: 1) rope-like structures, and 2) modeling and planning, shown in Figure 2.1. We will briefly review those subcategories. First, Section 2.1 will review manipulation tasks in terms of robots with flexible structures. We will describe the state-of-the-art of soft robotics and continuum manipulators. Second, we will briefly review coordinated manipulation for multi-robot systems such as multi-object manipulation, object towing via ropes, non-prehensile manipulation, and several rearrangement problems in Section 2.2. Section 2.3 will talk about tail robots, which use tail-like appendages to enhance robot agility. Section 2.4 will review motion

primitives, which are used for modeling complicated dynamic systems. To solve the object collection problem, we need a planner to sequence motion controls. In Section 2.5, we will review planners that can sequence motion primitives. In Section 2.6, we review motion planning problems with satisfiability modulo theories where solvers provide greater expressiveness and a high-level interface for expressing constraints in planning problems. These SMT-solvers dynamically incorporate constraints at the task level. Section 2.7 will talk about the estimation of system parameters for motion primitives. Since the estimation of system parameters is intractable in uncertain environments, we will review three typical methods.

2.1 Compliant, Underactuated Structures for Manipulation

In response to the desire to improve the versatility and grasping capabilities, many researchers of robotic manipulators have begun using compliant underactuated structures [13]. There are several reasons: robotic manipulators with flexible structures are much cheaper than an active controlled manipulators, and the flexibility can easily absorb impact forces to increase dexterous performance with easy control [14, 15, 16]. Due to the inherent advantages of flexible structures, robotic manipulators have been designed by the combination of soft materials and rigid plastic (or metal parts) to do complex tasks [17]. For example, if robotic arms are to work in close vicinity or in direct contact with fragile structures in the environment, the properties of compliance and deformation might be able to help to keep safe the target objects [18]. Our work is also motivated by the inherent advantages of the flexible structures, but our robots use these structures to manipulate objects.

Inspired by elephant trunks and snakes in the natural world, there has been much recent interest in continuum mechanisms, which are theoretically hyper-redundant, but in practice, underactuated systems [19, 20]. The main line of research is in the design and control of the continuum mechanisms with kinematic and dynamic models [20, 21]. These

hyper-redundant manipulators are highly controllable with complicated modeling. Walker [22] suggested an actively-controlled, long, string-like robot as a next-generation space robot, then Tonapi et al. [23], Walker et al. [24], Cohen et al. [25] proposed the design and the kinematic models of a string-like robot. Cowan and Walker [16] also suggested several possible dynamic motions, which may be feasible, such as using a flicking action to manipulate objects via the active controlled continuum robots. In our work, we are less concerned with specific models of flexible devices than with dynamic actions and their potential for interaction with the environment. The planning methods and models introduced in this dissertation allow both dynamic and quasi-static actions; the dynamic actions are shown to be efficient motions for some objectives (*e.g.,* object distance per action).

More broadly, the problem we study relates to prior efforts at manipulating objects without special purpose effectors, which includes work on pushing [26, 27], caging [28], striking [29], dragging [11, 30], or skimming [12]. Prior works controlled the rope motions kinematically, but our work uses the dynamics of the rope-like structures. Thus, we needed to consider the masses of the ropes, forces that lead to the motions, and tension induced via accelerations.

2.2 Coordinated Manipulation using Multi-robot Systems

Most efforts at manipulating objects with multiple mobile robots fall into one of two main categories based on the properties of objects being manipulated, either: (1) a large, heavy, or cumbersome object to be manipulated by multiple coordinated robots at a given time; or (2) a large number of distributed objects to be manipulated by multiple robots.

2.2.1 Manipulation of an Object by Multiple Coordinated Robots

Important early work in the first category includes: Rus et al. [31] demonstrated multiple robots cooperating in rearranging heavy furniture. Fink et al. [28, 32], Cheng et al. [33] caged large objects using multiple smaller sized mobile robots. Sartoretti et al. [34]

pushed one object using multiple surface vehicles. Kube and Bonabeau [35] considered the related question of cooperative transportation from the perspective of biology, providing examples of super-linear efficiency/capacity in several species of ant. In addition, more recently, Wilson et al. [36] presented an insect-inspired controller for cooperative transport by robots. The problem becomes more challenging when numerous objects must be manipulated by few robots. An appealing way to deal with this challenge was explored, first by Donald et al. [11], and by Bhattacharya et al. [30] more recently. They employ a pair of robots connected by a rope to manipulate multiple differently shaped objects, capitalizing on the rope's ability to adapt to the geometry of the objects and environment. We were greatly inspired by [30], but found that we needed to add several aspects (*e.g.,* a tail length and a load capacity) into our formulation so the plans would satisfy practical considerations.

Similarly, cooperative manipulations by towing objects via ropes are discussed by [37, 38, 39, 40, 41]. These rope-like structures are physically connected to objects, which are pulled by robots. Thus, the main problem is to balance dragging force among multiple robots. However, we are interested in robots using rope-like structures to enhance towing capacity with time. Sometimes, robots join the ends of their rope-like structure to encircle objects with greater towing capacity. Sometimes, individual robots fetch distant objects simultaneously. We need to balance the mix to maximize towing efficiency.

2.2.2 Manipulation of Distributed Objects by Multiple Robots

Within the second category, several researchers have shown how multiple robots can exploit parallelism to efficiently move objects spread throughout a large environment. Levihn et al. [42] studied what they term 'assignment space' to minimize task completion time—the same objective we minimize but we can also deal with others too. Fujii et al. [43], Oyama et al. [44], Inoue et al. [45] also considered robots that transport multiple ob-

jects simultaneously, allowing robots to grasp and drop multiple objects during sequential operation. An early, under-appreciated paper, Yamashita et al. [46] proposed the use of tools to allow the simultaneous manipulation of multiple objects by multiple robots, including rope and cords. Our present work contains some of these elements, but deals with a distinct problem in Section 4: (1) Our robots can form a sub-team to maximize efficiency operating as a tightly-knit pair or separately; (2) We can solve a variety of different object collection tasks.

The motion planning aspects of multi-objects collection via cooperative towing fall within the class of vehicle routing problems [47]. Ralphs et al. [48] formulated a discrete optimization routing problem and computed a solution using a sub-optimal heuristic which subdivides the overall optimization into subproblems; the approach we employ can be said to be in the same spirit. Similarly, Mathew et al. [49] proposed a heterogeneous multi-robot delivery problem as a discrete optimization problem, and solved it using a reduction from the Traveling Salesman Problem. Coltin and Veloso [50] introduced the pickup and delivery problems, and solved it by using auction-based scheduling algorithms that can transfer items between robots. More generally, multi-robot path planning problems are discussed by [51, 52, 53]. Luna and Bekris [51] proposed an efficient algorithm that can swap positions of any other robots until robots can navigate to goal positions. van den Berg et al. [53] addressed centralized path planning that can partition the robots to minimize a composite of all robots. Turpin et al. [54] presented the optimal trajectory planning for interchangeable robots. Wagner et al. [55] and Wagner and Choset [56] proposed sub-dimensional expansion to find optimal paths at low computational cost, and Solovey and Halperin [52] addressed the k-color planning problem, and proposed an algorithm that can solve general cases of multi-robot motion planning problems. We also address a discrete multi-robot motion planning problem, but our method can cope with a variety of challenging scenarios.

In terms of coordination, we can think of physically joining motions: docking and rendezvous motions. Docking and rendezvous planning have been studied for several decades. Docking (or rendezvous) to charge stations is a critical problem in the contexts of long-term missions performed by multiple robots [57]. Roh et al. [58] demonstrated robust docking mechanisms via magnetic force. Our previous work [59] showed inter-vehicle docking via visual servoing on the surface water. In this dissertation, our robots need inter-tail docking to work on cooperative missions similar to [12, 30]. This inter-tail docking allows two tail robots to link tails, and then the conjoined tail robots are treated as tethered robots.

2.3 Tail Robots

Recently, tails and tail-like structures have been used only in limited settings and primarily to enhance a robot's agility. Several researchers [60, 61, 62, 63] have shown a lizard-inspired active tail that can enable the robot to leap with stabilization over broken rubble. The elasticity of the active tail gives a fast response for stabilization on rough terrains [64]. A dynamic active tail makes rapid turning at high-speed possible for robots [65, 66, 67, 68, 69]. Unlike prior work, our work [70] was focused on object manipulation via the dynamics of the compliant tail and its potential for interaction with its environment.

2.4 Motion Primitives

The work discussed in the preceding sections mostly focused on specific kinematic and dynamic models of complicated flexible structures. However, our study is concerned with how to use the dynamics of the compliant tail and its potential for interaction with its environment. Motion primitives are well known methods to model complicated system and have a merit in terms of simplicity, stability, and robustness to achieve a specific task [71]. Jenkins and Matarić [72] proposed a data-driven approach for deriving human-motion primitives from time-series data of human motion. Hauser et al. [73] used a small

set of high-quality motion primitives that have been generated off-line, and found a path using sampling strategy for a probabilistic, sample-based planner for several different terrains. Powell et al. [74] presented an approach dealing with multiple motion primitives for walking and stair climbing by using the examination of experimental human data. We also use a data-driven approach to get characteristics of primitives.

Motion primitives are employed to propagate forward motions in the Rapidly-exploring Random Tree (RRT), called kinodynamic RRT [75]. State lattices (Lattice-based graphs) are constructed of simple motion primitives connecting one state to another [76]. Butzke et al. [77] introduced augmented lattice-based path planning by using controller-based motion primitives that can outperform in specialized regions such as GPS-denied areas. Our approach is similar to Butzke et al. [77]. We have a set of stable motion primitives that can be selected considering preferences and environmental settings.

There exist many applications using motion primitives to overcome the difficulties of modeling and planning in different settings. Vonásek et al. [78] used simple motion primitives to reduce computational time for modular robots. Krontiris and Bekris [79] used different types of motions corresponding to the stable transit and transfer actions for an object rearrangement problem. Gray et al. [80] employed parameterized motion primitives to enhance agile drifting maneuvers. Paranjape et al. [81] applied motion primitives for fast flight through a forest. Gupta et al. [82] used pick/drop, spread, and tumble motion primitiives for object sorting in cluttered environment. In our studies, motion primitives also help to reduce model complexity, and they allow new tasks.

2.5 Planning under Motion Uncertainty

Although motion primitives enable modeling complicated systems, and then allow for a variety of manipulation tasks, they also require sophisticated sequencing to execute given tasks. The most popular framework for planning under motion uncertainty is the Markov

Decision Process (MDP). The MDP works very well in discrete state spaces of moderate size. For manipulation, it is non-trivial (or infeasible) to build analytical models that capture the full complexity of the object-tool-environment interactions involved. The inevitable consequence is uncertainty, which must be dealt with in some way. Dogar and Srinivasa [83] reduced the uncertainty of a pushing action by utilizing the funneling effect of pushing. Related ideas include that of Meriçli et al. [27] who proposed an experience-based approach that uses past motions, and Phillips et al. [84] developed an online motion planning approach that makes use of information from previous searches. For motion planning in continuous space, sampling-based motion planning techniques have received much attention over the past 15 years. In particular, the Rapidly-exploring Random Tree (RRT) operates by growing a tree in state space, repeatedly sampling new states by picking the closest existing node and steering towards that sample [85]. Berenson et al. [86] demonstrated repair of paths from the past-learned paths using the RRT. This study uses the RRT framework using manipulation primitives and, similar to Bry and Roy [87], motion errors are propagated using a particle-based representation.

2.6 Planning with Satisfiability Modulo Theories

Satisfiability Modulo Theories (SMT)-solvers [88, 89, 90, 91] are fully automatic satisfiability checkers for a set of logical formulas, where the SMT is a combination of various theories represented as formulas in first-order logic. This allows a powerful expressive and high-level interfaces for representing constraints in robotics research [92]. SMT-solvers explore the large combinatorial search space with considerable improvements in the scalability of state space [92]. Recently, several people [93, 94, 95] have been introduced for combining task and motion planning by SMT-solvers. Hung et al. [93] used SMT-solvers for motion planning with rectangular obstacles. Nedunuri et al. [94] integrated task and motion plans into SMT-solvers. Dantam et al. [95] extended task and motion planning

problems with incremental solving by adding and deleting constraints via SMT-solvers. Saha et al. [96] introduced SMT solvers to synthesize trajectories by composing a set of motion primitives. Our work also use the scalability and flexibility of SMT-solver's properties in multi-robot motion planning problems for collecting multiple objects. We construct logical formulas to represent various goal constraints so that we can handle quite general object collection problems.

2.7 Estimating System Parameters

In practice, estimating system parameters is very important to control systems performing tasks under uncertainty. There are several methods for system identification: analytical models, physics-based simulations, and "calibration experiment of system" are employed. Physics-based simulators are well known methods to predict physical interactions in various complicated tasks [97, 98, 99]. There are various kinds of data-driven approach: especially, reinforcement learning has been used for self-improvement of primitives for several decades [100, 101, 102, 103]. Mahadevan and Connell [100] first showed new behaviors are learned by trial and error using a performance feedback function. Soni and Singh [102] demonstrated transportation of a ball by a robot, which was learned with semi-automatically discovered options. Kolter and Ng [104] showed a local policy search for learning a jumping behavior for a robot dog. Kober et al. [105] employed non-parametric regression approaches in order to adjust the existing behaviors of a lower-level controller for dart throwing and table tennis hitting tasks. Daniel et al. [103] executed a learning method to sequence motion primitives with policy search. Lastly, Michels et al. [106] demonstrated a task of driving a vehicle at high speeds through unstructured outdoor environments. This work [70] also uses a physics-based simulator and data-driven approach to estimate system parameters.

3. USING A COMPLIANT, UNACTUATED TAIL TO MANIPULATE OBJECTS*

3.1 Introduction

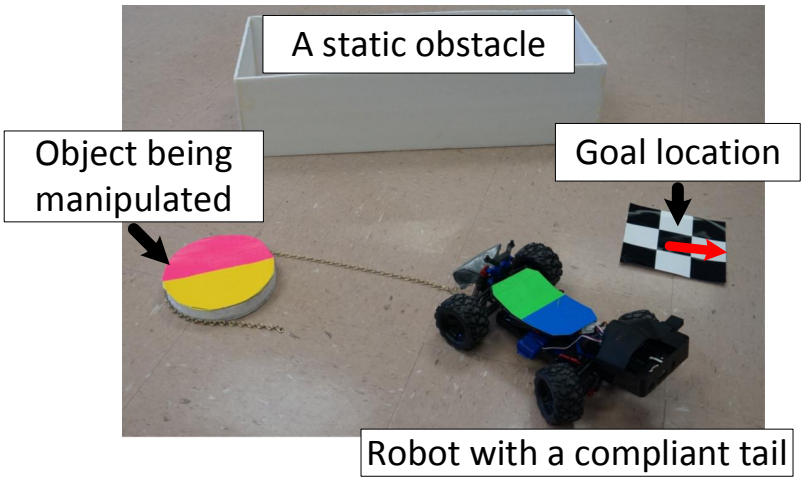


Figure 3.1: An example scenario: the robot's task is to move the object from the initial pose to the goal. The novel idea in this work is that the robot only uses its tail to translate and rotate the object.

This chapter investigates how a robot may use flexible rope-like structures as tools in diverse ways, demonstrating high-speed dynamic (*e.g.,* striking) and high-precision quasi-static (*e.g.,* dragging) actions, as well as mixtures thereof. Our novel robot system with a flexible rope-like structure attached as a tail is shown in Figure 3.1. Four main challenges arise in using this system for manipulation. We mentioned those challenges in Chapter 1.2. This chapter details an approach that addresses these challenges.

*Reprinted, with permission, from Young-Ho Kim and Dylan Shell, "Using a compliant, unactuated tail to manipulate objects," IEEE Robotics Automation Letters ©2017 IEEE [70].

We began by constructing a set of primitives that, while parsimonious, possesses sufficient richness to enable useful manipulation. Associated with each primitive motion is a forward model that is used to predict the object's subsequent position and orientation within the workspace. Finally, a sequence of primitive motion actions are sought to solve a given manipulation problem instance. We employ a sampling-based motion planning technique coupled with a particle-based representation to find such sequences.

3.2 Preliminaries and Problem Description

Let M_o, M_r, and M_t represent the mass of the object, the robot, and the tail, respectively. We consider three coefficients of friction μ_o, μ_r, and μ_t for the object, robot, and tail, in their contact with the workspace floor. One additional coefficient of friction, μ_{o-t} is that between the object and tail. For a tail of length L, the physical system parameters form a tuple $p = \langle M_o, \mu_o, M_r, \mu_r, M_t, \mu_t, L, \mu_{o-t} \rangle$. The object's state $X_o = (x, y, \theta) \in \chi$, has (x, y) as the position of the center of the object in $\mathbb{R}^2$ and θ its orientation. The robot's state X_r is defined analogously. We let $\tilde{X}_t$ be a vector describing the tail configuration, either as an approximation via n segments, or a parametric function. Then the manipulation system can be treated as a transition function F. Using control $u \in U$, F makes the transition from $X(k)$ to another state $X(k+1)$:

$$X(k+1) = F(X(k), u(k); p), \tag{3.1}$$

where X contains states of X_o, X_r, and $\tilde{X}_t$.

3.2.1 Simplifying Assumptions

Our robot has an inelastic tail that it uses for manipulating a cylindrical object, and the robot is assumed to be powerful enough to drag the tail along with the load of the object. Also, we treat the robot and the object states as observable, while the tail configurations

are not.

3.2.2 Problem Definition

Let $\mathcal{J}(X) = (J_1(X), \ldots, J_N(X))$ be a vector-valued cost function, where the integer $N \geq 1$ is the number of objectives that describe aspects of the system's performance. For instance, J_i could correspond to a measure of path safety, a measure of path accuracy, execution time, or navigation cost.

Problem: Given U and p, manipulate the object from the initial pose $X_S = (x_s, y_s, \theta_s)$ to the target pose $X_G = (x_g, y_g, \theta_g)$, making all state transitions via the robot tail, finding a sequence of motions $\pi^* = [u_1^*, \ldots, u_N^*]$ that minimize $\mathcal{J}(X)$, constructed as a linear combination of the individual objectives

$$\mathcal{J}(X, \vec{\alpha}) = \sum_{i=1}^{N} \alpha_i \cdot J_i(X), \tag{3.2}$$

for some given $\vec{\alpha} = \alpha_{1,\ldots,N} \in \mathbb{R}^+$.

The total cost function for a trajectory $[X_0, \ldots, X_N]$ is

$$\mathcal{J}_{([X_0,\ldots,X_N],\vec{\alpha})} = \sum_{i=0}^{N} \mathcal{J}(X_i, \vec{\alpha}). \tag{3.3}$$

So the problem is to find

$$\pi^* = \underset{[X_0,\ldots,X_N]\in\chi}{\operatorname{argmin}} \mathcal{J}_{([X_0,\ldots,X_N],\vec{\alpha})}, \tag{3.4}$$

where $\vec{\alpha}$ represents preferences over objectives, and N is the number of waypoints.

The overall system architecture appears in Fig 3.2.

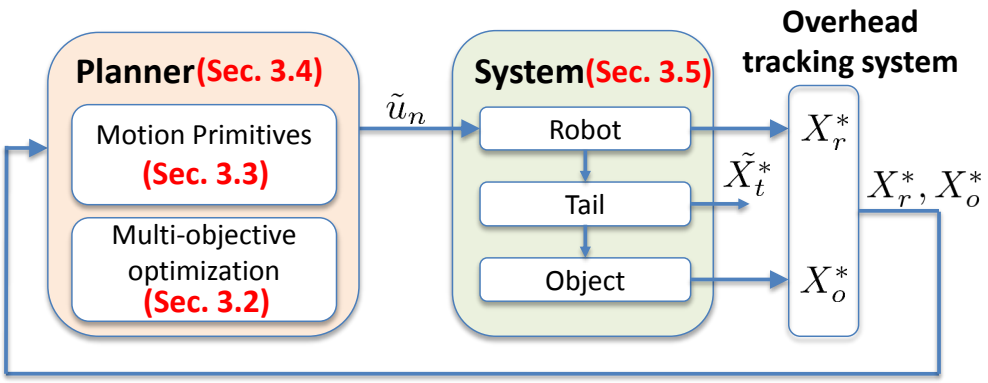


Figure 3.2: An overview of the system showing sections with discussion.

3.3 Modeling Motion Primitives

To simplify both the modeling and planning problems, we use a small set of motion primitives: each represents a simple and stable activity which is, ideally, helpful in achieving a goal. We desire a small portfolio of motion primitives, each contributing some aspect lacking in the others. Additionally, an important factor is each primitive's amenability to modeling and, ultimately, its predictability; this is a function of primitive complexity. Three motion primitives, broadly representative of three broader classes, are considered: 1) quasi-static motion primitive — modeled as deterministic, 2) quasi-static motion primitive involving stick-slip frictional transitions — modeled as non-deterministic, and 3) dynamics-based motion primitive where the tail strikes the object — which is non-deterministic. From 1) to 3), the tractability and accuracy of modeling decrease owing to the inherent complexity of the physical phenomena involved. Our philosophy is to profit from diversification: we resort to a data-driven approach when the system is highly non-linear and transitions are too complicated for us to treat otherwise.

We consider a set of motions, each of which represents a sequence of controls, u, executed over an interval of time, which results in a transition between two states in the

continuous state space. Each of these is characterized off-line, either through analytical models, simulation, or calibration experiments (as will be described in Section 3.3.4). We also allow primitives to be parametrized, exploiting regularity and symmetries in doing so. Let $\tilde{u}(\phi)$ denote a set of motions, where we have expressed the fact that they are parameterized by angle ϕ, representing the direction in which to move the object. Then we define a motion primitive $\tilde{\mathcal{U}}(\phi)$ to be a tuple $\langle \tilde{u}^-(\phi), \tilde{u}(\phi), \tilde{u}^+(\phi) \rangle$, where $\tilde{u}^-$ represents an initialization motion, $\tilde{u}$ represents a motion moving the object, and $\tilde{u}^+$ represents a termination motion. The following sections discuss these in further detail.

First we note that we make two design decisions:

Design Decision 1 *We do not represent the tail explicitly.*

Rationale: This vastly simplifies the planning and representation problems. Our approach is to have each motion primitive $\tilde{\mathcal{U}}$ include an 'initialization' motion $\tilde{u}^-$, which normalizes the tail configuration, laying the tail out into some predictable configuration no matter its prior condition. This allows $\tilde{\mathcal{U}}$ to follow any state. In practice $\tilde{\mathcal{U}}$ is imperfect so the uncertainty of the tail configuration is implicitly included in the parameterized motion primitives, as discussed later.

Design Decision 2 *We assume the robot motion is deterministic, whereas we must treat the object motion as non-deterministic for any* $\tilde{\mathcal{U}}$.

Rationale: The robot state and the object state are coupled via the tail. It is possible to estimate the robot state relative to the object state if the robot's control policy seeks to maximize this form of information. For example, the robot can stop the instant the object stops moving. Then, given $\tilde{\mathcal{U}}$, the object state can be used to help determine the relative location of the robot, a step which helps reduce planning complexity. There are small errors in pose estimates in practice, but are incorporated in the motion primitive model easily.

These two design decisions yield a simplification of Equation (3.1) to give a transition function F_s as

$$X_o(k+1) = F_s(X_o(k), \tilde{\mathcal{U}}(k), p), \tag{3.5}$$

wherein we only need consider the object's state, because the robot state is determined by the object state.

3.3.1 Robot Motion Model

The robot has motion constraints that affect the feasible tail configurations and, ultimately, the object motions. Here we assume a simple car model for the robot. We generate the robot trajectories via Dubins paths that give the shortest path in the two-dimensional Euclidean space, where paths consist of circular curves of maximum curvature and linear motions. The paths are also used in obstacle environments [107]. All robot motions generated in this dissertation are made by Dubins paths, including atomic actions in each $\tilde{\mathcal{U}}$.

3.3.2 Quasi-static Model: Simplified Analytic Model for Dragging Motions

First, consider a dragging motion where the tail and object make contact throughout: The robot approaches the object's side, wraps the tail around the object, and moves forward, resulting in simultaneous rotation and translation of the object. There is no need for explicit consideration of the tail's configuration other than basic physical properties such as its mass and coefficients of friction. The following analysis considers the quasi-static regime, assuming small inertial forces compared to the tail's contact forces.

A free body diagram for this scenario is shown in Figure 3.3. Suppose that the direction of the object's translation is in the horizontal x-direction (i.e., for $\phi = 0$). The basic dragging motion depends on tensions T_L and T_R, on the left and right sides of the tail, where the object is taken as the dividing point. We denote the mass of the left hand side of the tail

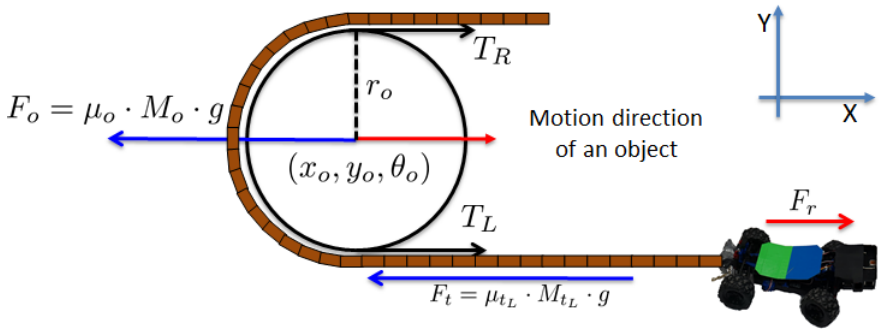


Figure 3.3: A free body diagram for the dragging motion.

by M_{t_L}, and the force induced by the robot by F_r. While dragging, F_r may increase as M_{t_L} increases, causing an increase in the left side of the tail's static friction F_t. Still, $F_r - F_t$ is constant when the robot's force F_r increases (e.g., via a feedback controller) to ensure the quasi-static motion is maintained. Equation (3.6) and Equation(3.7) are applications of Newton's second law, here for translation in the x-direction. When the right side of the tail is not moving due to friction with the floor, the operation is identical to the physics of an ideal pulley, permitting us to write $T_L = T_R$ in Equation (3.7). The object exerts the total frictional force of $T_L + T_R = 2T$, meaning that if the robot moves distance of $2d$, the object moves distance of d.

$$T_L = F_r - F_t, \tag{3.6}$$

$$T_L + T_R = F_o. \tag{3.7}$$

Ideal pulley physics is only applicable when friction fixes the right side of the tail. As the robot drags its tail, the proportion of mass on the right (left) side decreases (increases,

respectively). Eventually T_R cannot be sustained; the friction breaks down when the mass on the tail's right side is inadequate, which is dependent on the length of the tail. Let the minimum length of the right side of the tail be L_{min}, then there are two distinct frictional phases: 1) the right position of the tail stays while the left position moves; 2) the whole tail moves, slipping over the floor. For the first phase, we can get a closed-form solution as it is identical with a pulley physics system.

Focusing on this first phase of motion, we call this our fine motion primitive, $\tilde{\mathcal{U}}_0$,

$$\begin{aligned} X_{o(x,y)}(k+1) &= d\,A + X_{o(x,y)}(k), \\ X_{o(\theta)}(k+1) &= X_{o(\theta)}(k) + \frac{2d}{r}, \\ X_{r(x,y)}(k+1) &= 2d\,A + X_{r(x,y)}(k), \end{aligned} \tag{3.8}$$

where d is a limited distance, depending on L, since the robot can move distance at most $L - L_{min}$. As before, ϕ denotes the direction of motion, but r is the object radius, and A is $[\cos\phi, \sin\phi]$.

The uncertainty in this motion primitive has its origins in the robot's motions. The robot executing $\tilde{\mathcal{U}}_0$ must ensure that it operates in the first frictional phase only. This requires that it stop during the dragging motion and then separate the tail from the object. This additional step (a termination step, $\tilde{u}_0^+$) is fairly complicated, but *is the price demanded for a primitive that realizes such a simple model*.

A second, more cavalier, primitive has the robot simply continue forward even after the tail and object begin to slip (in Figure 3.3, it keeps going in the positive x-direction). Passing through the first frictional phase into the second, a highly non-linear transition from sticking to slipping occurs manifesting itself as additional uncertainty for this primitive $\tilde{\mathcal{U}}_1$.

The second dragging primitive, $\tilde{\mathcal{U}}_1$, has a pair of stochastic state transition functions:

$$\begin{aligned} X_o(k+1) &= X_o(k) + \omega_1, \quad \omega_1 \sim \mathcal{N}(X_{\mu_1}, \Sigma_1), \\ X_r(k+1) &= X_o(k+1) + \omega_2, \quad \omega_2 \sim \mathcal{N}(X_L, \Sigma_2), \end{aligned} \tag{3.9}$$

where X_{μ_1} and Σ_1 are the mean and variance of the object state transitions via $\tilde{\mathcal{U}}_1$. Here X_L and Σ_2 describe the robot's state error relative to $X_o(k+1)$, parameterized by ϕ. The robot is dragging the object initially, but at some point the tail begins to slip, and eventually friction causes the object to come to rest. Our controller ensures that the robot stops then too. Thus, the robot's state distribution depends on the object location, which facilitates planning because the prediction of the robot state can be determined relative location of the object (and ϕ, and the tail length).

3.3.3 Dynamic Model: High-Speed Striking

In addition, we investigated a high-speed primitive motion that has the robot exploiting the dynamics of the tail to lash the object. The robot approaches the object from an angle dependent on ϕ, then drives with constant velocity and at the maximum steering angle while keeping a short separation distance. As the tail is moved at high speed, its internal tension stiffens the rope along its extent, and when the object is struck both rotation and translation result.

The striking primitive, $\tilde{\mathcal{U}}_2$, has state transition function with a probability distribution:

$$\begin{aligned} X_o(k+1) &= X_o(k) + \omega_3, \quad \omega_3 \sim \mathcal{N}(X_{\mu_3}, \Sigma_3), \\ X_r(k+1) &= X_o(k+1) + \omega_4, \quad \omega_4 \sim \mathcal{N}(X_{any}, \Sigma_4), \end{aligned} \tag{3.10}$$

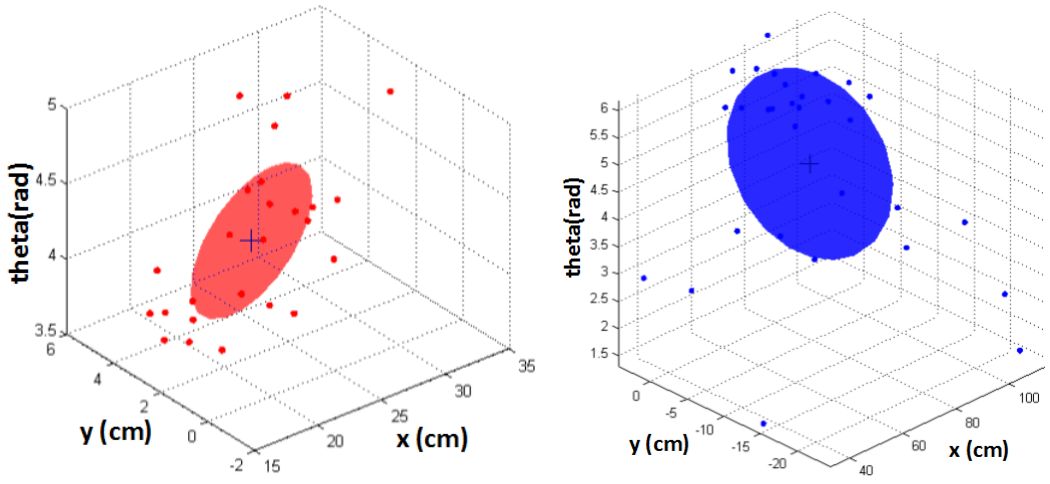
where X_{μ_3} and Σ_3 are the mean and variance of the object state transitions via $\tilde{\mathcal{U}}_2$. The mean and variance of the relative robot state is given by X_{any} and Σ_4, which capture dependency on ϕ, and intrinsic robot control errors.

3.3.4 Learning Model Parameters

Specific learning model parameters of X_{μ_1}, X_{μ_3}, Σ_1, Σ_2, Σ_3, and Σ_4 are reported here.

Our motion primitives have several parameters. Primitive $\tilde{\mathcal{U}}_0$ has an analytical model (but does include a small variance). For $\tilde{\mathcal{U}}_0$, we only need to know L_{min}, which is approximately $20\ cm$ in our tests. However, $\tilde{\mathcal{U}}_1$ and $\tilde{\mathcal{U}}_2$ are much more complicated, so we resorted to collecting data. We placed the robot and tail in random initial configurations, then we executed the primitives over 20 times while recording data.

The following summarize the collected parameters through the repeated execution of each motion primitive in Figure 3.4.



(a) $\tilde{\mathcal{U}}_1$ (b) $\tilde{\mathcal{U}}_2$

Figure 3.4: The object is located at (0,0,0), and the robot executes each motion primitive with $\phi = 0$. Settings match those in Figs 3.6 and 3.7.

Motion Primitives	$\tilde{\mathcal{U}}_0$	$\tilde{\mathcal{U}}_1$	$\tilde{\mathcal{U}}_2$
Properties	Statics-based idealized & interaction	Statics-based stick/slip together	Dynamics-based
Model	Deterministic w/ small variance added	Stochastic	Data-driven (Stochastic)

Figure 3.5: A diverse set of motion primitives are used. From left to right, the object travel distance per unit of time increases while the model tractability and model accuracy decrease.

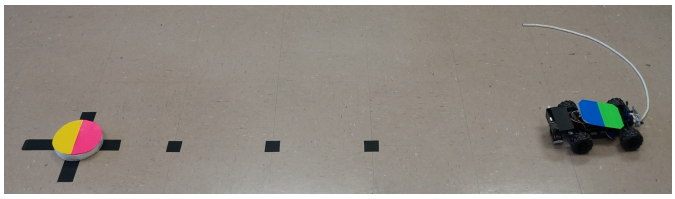
$$X_{\mu_1} = \begin{bmatrix} 26.9 \\ 5.23 \\ 4.899 \end{bmatrix} \quad \Sigma_1 = \begin{bmatrix} 5.1673 & 1.6611 & 0.2215 \\ 1.6611 & 1.4337 & 0.1414 \\ 0.2215 & 0.1414 & 0.0183 \end{bmatrix}$$

$$X_{\mu_3} = \begin{bmatrix} 80.04 \\ -4.57 \\ 4.51 \end{bmatrix} \quad \Sigma_3 = \begin{bmatrix} 122 & 5.6 & 0 \\ 5.6 & 6.99 & 0.902 \\ 0 & 0.902 & 0.4183 \end{bmatrix}$$

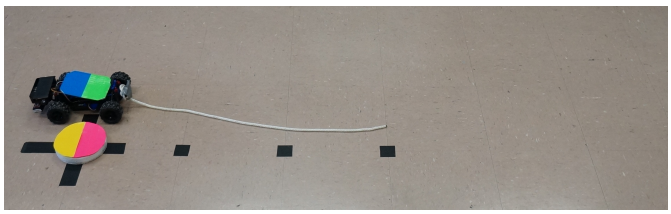
Σ_2 and Σ_4 are taken as $[3, 0.1, 0]$, which are measured directly via execution of Dubins paths.

3.3.5 Recap of Motion Primitives: Initialization

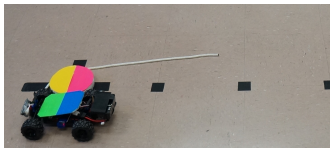
The motion primitives require initialization motions, $\tilde{u}^-$, to ensure the tail is in a well-defined configuration. Some motion primitives also need additional actions, $\tilde{u}^+$, to simplify the robot's state estimation. This section reviews the motion primitives, detailing these additional motions which have not yet been described. We summarize the set of primitives we use in Figure 3.5: from left to right tractability and accuracy of modeling decrease because the physical phenomena involved are complex. Data-driven models are used when the system is highly non-linear and transitions are too complicated to treat otherwise. Though not shown, primitive $\tilde{\mathcal{U}}_2$, has merit in terms of the object travel distance per unit of time.



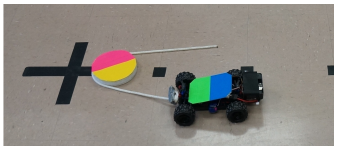
(a) T= 0 sec.



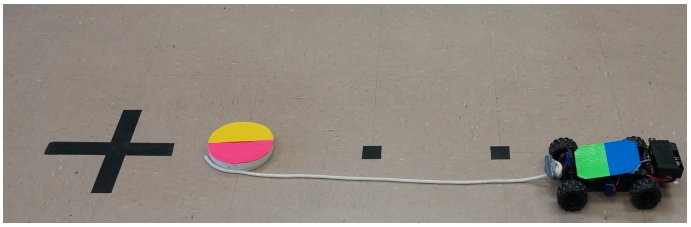
(b) T= 6 sec.



(c) T= 12 sec.



(d) T= 20 sec.



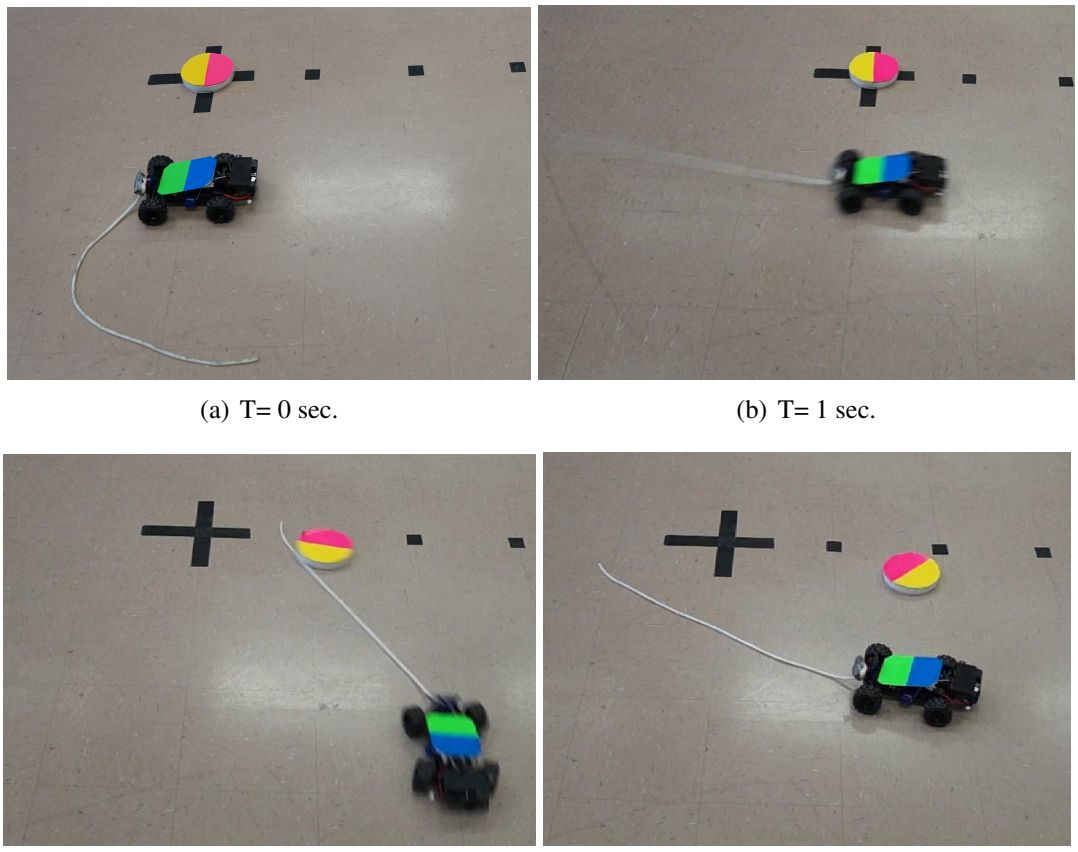
(e) T= 25 sec.

Figure 3.6: The initialization motion for $\tilde{u}_0^-$ and $\tilde{u}_1^-$ is from (a) to (b). Then, the robot drags its tail to move the object through (c), (d) and (e). If the robot wants to execute $\tilde{\mathcal{U}}_0$ only, the robot might stop execution as shown in (d). For $\tilde{\mathcal{U}}_1$, the robot keeps dragging until the object stops moving, shown in (e).

The photos in Figure 3.6 show $\tilde{\mathcal{U}}_0$ and $\tilde{\mathcal{U}}_1$. There are four phases. The first sets the robot and object some distance (and bearing ϕ) apart—shown in Figure 3.6(a). Then, the robot executes a pre-planned path, $\tilde{u}^-$, bringing the robot next to the object, as in Figure 3.6(b). Next, in Figure 3.6(c), the robot makes a simple surrounding motion that wraps the object, whereafter the robot drags the tail for distance of $2\,d$. For $\tilde{\mathcal{U}}_0$, the robot will stop in Figure 3.6(d). For $\tilde{\mathcal{U}}_1$, the robot will continue until the object stops moving in Figure 3.6(e).

For $\tilde{\mathcal{U}}_0$, the robot reverses over its tail to separate the tail from the object after the object has stopped. These additional motions, $\tilde{u}_0^+$, involve execution of a pre-planned path. These are not needed for $\tilde{\mathcal{U}}_1$, thus $\tilde{u}_1^+ = \varnothing$.

Figure 3.7 shows the three phases of $\tilde{\mathcal{U}}_2$. Initially, the robot navigates to the initialization location via $\tilde{u}_2^-$, shown in Figure 3.7(a). Second, the robot executes its high-speed motion with steering at hard lock. The tail configuration is predictable as it essentially becomes a semi-rigid body—Figure 3.7(b). Next the tail hits the object, shown in Figure 3.7(c), and the object moves. Finally, to be consistent with the model, the robot moves to a location relative to the object's resting pose, which is shown in Fig 3.7(d). This last motion is $\tilde{u}_2^+$.



(a) T= 0 sec. (b) T= 1 sec.

(c) T= 1.5 sec. (d) T= 2 sec.

Figure 3.7: (a) The initialization, $\tilde{u}_2^-$, positions the robot relative to the object. (b) This shows a high-speed motion. The robot makes a circular motion. (c) The tail configuration is a semi-rigid body after first round, and then the tail hits the object. (d) The robot stops at some location relative to the object ($\tilde{u}_2^+$).

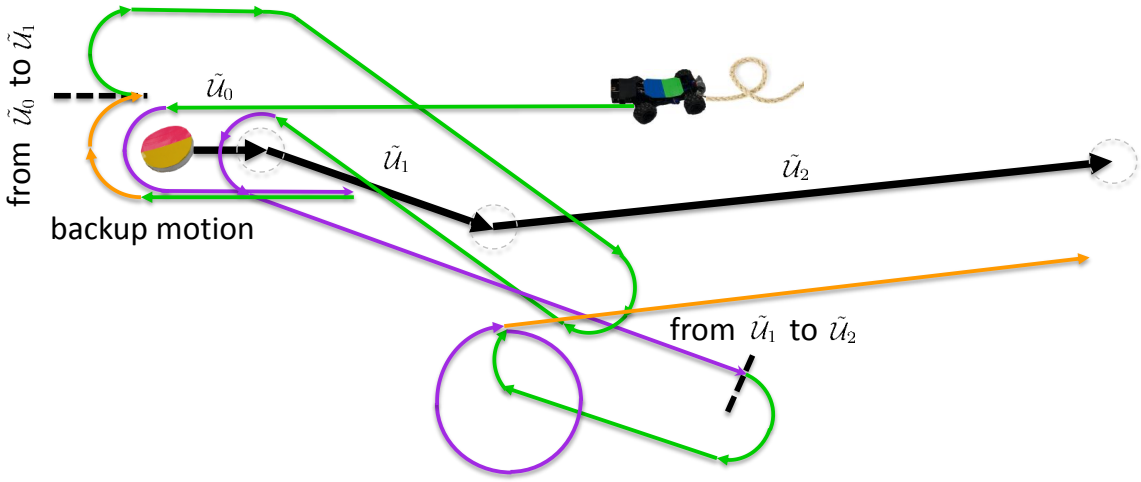


Figure 3.8: Motion primitives are sequenced together. The black line shows segments of motion that the object undergoes. The green lines are the initialization motions, $\tilde{u}^-$, and the orange solid lines are termination motions, $\tilde{u}^+$. The purple lines are portions of the path which move the object. The broken black lines show points of transition between primitives.

3.4 Planning with the Motion Primitives

Having outlined the primitives individually, the next step is to plan and execute sequences of motion primitives. Figure 3.8 provides an illustration of how to sequence motion primitives as an example: The path consists of the sequence $\tilde{\mathcal{U}}_0$, $\tilde{\mathcal{U}}_1$, and $\tilde{\mathcal{U}}_2$. First, the robot executes $\tilde{\mathcal{U}}_0$. The robot path is generated via a Dubins curve and the robot follows the green line as its initialization, $\tilde{u}_0^-$. It then makes the motion shown by the purple line to move the object. At some point, the robot stops, executes a backing up motion as $\tilde{u}_0^+$ to separate the tail from the object. Here a transition from $\tilde{\mathcal{U}}_0$ to $\tilde{\mathcal{U}}_1$ occurs. The robot moves along the (green) initialization path as $\tilde{u}_1^-$ and then the robot executes the dragging motion (purple). The final transition is from $\tilde{\mathcal{U}}_1$ to $\tilde{\mathcal{U}}_2$. The robot initializes with $\tilde{u}_2^-$ and then makes a high-speed circular motion. Once the object stops moving, the robot goes to its final location via $\tilde{u}_2^+$.

How does a planner find such a sequence, especially since the primitives include motion uncertainty? We use the Rapidly-exploring Random Tree (RRT) [75] and add a particle-based representation for uncertainty of the object's state transitions. The algorithm operates on a graph describing the object's state space. Each vertex has the object's current state, X_o and the robot current state X_r, and the belief states of each object state have a weighted set of N particles, $\{(X_{o_1}, w_1), \ldots, (X_{o_N}, w_N)\}$. Each edge is labeled with the motion primitive needed to traverse between the associated vertices for the object and the robot trajectories in Sec. 3.3.1.

Algorithm 1 RRT_PLANNING

INPUT: X_S, X_G, robot initial location X_{r_0}, initial tree $\tau_0(X_S)$
OUTPUT: $\pi^* = (\tilde{\mathcal{U}}_1^*, \ldots, \tilde{\mathcal{U}}_N^*)$
TREE_BUILDING_PHASE(X_S, X_{r_0}, $\tau_0(X_S)$)
SEARCH_PHASE(τ, $\vec{\alpha}$, X_G)

Algorithm 2 TREE_BUILDING_PHASE(X_S, X_{r_0}, $\tau_0(X_S)$,)

INPUT: $[\tilde{\mathcal{U}}_0, \ldots, \tilde{\mathcal{U}}_N]$, X_S, robot initial location X_{r_0}, an initial tree $\tau_0(X_S)$,
OUTPUT: τ_K.
for k = 1 to K **do**
  $(X_{rand}, \tilde{\mathcal{U}}_{rand}(\phi))$ = SAMPLE()
  X_{near} = NEAREST(X_{rand}, $\tilde{\mathcal{U}}_{rand}(\phi)$, τ_k)
  if NEW_STATE(X_{rand}, $\tilde{\mathcal{U}}_{rand}(\phi)$, X_{near}, X_{new}) **then**
    $X_{new}.particles$ = PROPAGATE(X_{near}, $\tilde{\mathcal{U}}_{rand}(\phi)$)
    τ.add_vertex(X_{new}, $\tilde{\mathcal{U}}_{rand}(\phi)$)
    τ.add_edge(X_{near}, X_{new}, $\tilde{\mathcal{U}}_{rand}(\phi)$)
  end if
end for

Algorithm 3 $X_{new}.particles$ = PROPAGATE(X_{old}, $\tilde{\mathcal{U}}, \phi$)

1: **for** $p = 1$ to P **do**
2: $X_{new}.particles_p = X_{old}.particles_p$ + SAMPLE($\tilde{\mathcal{U}}, \phi$)
3: UPDATE_WEIGHT($X_{new}.particles_p$)
4: $X_{new}.particles_p.cost$ = UPDATE_COST($X_{new}.particles_p$)
5: **end for**

Algorithm 1 gives the RRT algorithm with a mechanism adapted for propagation of errors. First, we build the tree with our motion primitives from the object's start node. Then we search for a low-cost path based on the scalarized cost function. (In our implementation these procedures are performed offline.) Algorithm 2 consists of following basic functions: The SAMPLE function returns uniform samples of the object state X_o in χ_{free}, which is the collision-free space (x, y, θ). We also sample a heading angle $\phi \in [0, 2\pi)$ and a specific motion primitive $\tilde{\mathcal{U}}$ with object rotation direction. The NEAREST function finds the nearest node X_{near} from τ_k. Then, the NEW_STATE function determines the new tree node based on global constraints, such as being collision-free (including the object state transitions and the robot trajectories). When new nodes are added, we propagate object motion errors. In Algorithm 3, we compute a new particle state by sampling particles via the probability distribution of the motion primitive. Then we update the weights, normalizing them based on the average of $X_{new}.particles$, also updating $\mathcal{J}(X)$.

Once we have a random tree, the next step is to find a sequence of motion primitives via the SEARCH_PHASE in Algorithm 1. We use Dijkstra's algorithm with $\mathcal{J}(X_i, \vec{\alpha})$ in Algorithm 4.

Algorithm 4 SEARCH_PHASE(τ, $\vec{\alpha}$, X_G)

INPUT: τ, $\vec{\alpha}$, object goal location X_G
OUTPUT: $\pi^* = (\tilde{\mathcal{U}}_1, \ldots, \tilde{\mathcal{U}}_N)$
cost$_{init}$ = 0
Q = X_{init} and ClosedSet = {}
while Q $\neq 0$ **do**
    $X_{current}$ = POP(Q)
    ClosedSet.add(current)
    for each neighbor of $X_{current}$ **do**
        $\mathcal{J}_{X_{neighbor}} = \mathcal{J}(X_{neighbor}, \vec{\alpha})$
        Q = Q $\cup X_{neighbor}$
    end for
end while

3.5 Experimental Results

This section presents results from several physical robot experiments. The experiments we report constitute a demonstration of manipulation through the use of an unactuated tail, providing proof of sufficiency, and showing that though our approach involves simplifications, it nevertheless enables planning and successful execution of actions to manipulate objects within the workspace. The work also provides a basis for exploring the importance of several notions of path cost, complex preferences between objectives, and how a diverse portfolio of primitives enables synergy.

3.5.1 System Setup

We used an RC car controllable to velocities between $0.4\ m/s$ for $\tilde{\mathcal{U}}_0$ and $\tilde{\mathcal{U}}_1$, and over $1.5\ m/s$ for $\tilde{\mathcal{U}}_2$. An embedded computer on the car controls the robot and communicates with a separate computer integrated with an overhead tracking system which localizes the object and the robot. The software uses the ROS framework. All experiments are conducted in our test arena of size $5.18\ m \times 4.27\ m$. Pose errors are estimated to be $\pm\ 5\ cm$ and $\pm\ 10\ rad$, respectively. The tail configuration was not tracked. The following physical properties were measured: the target object is a cylinder with $mass = 30\ g$, $\mu_o = 0.6655$,

and $radius = 7.5\ cm$; the robot weighs $700\ g$; tail is a chain with $mass = 35\ g$, $length = 70\ cm$, and $\mu_t = 0.5952$.

3.5.2 Scenarios: Planners, Environments, and Objectives

3.5.2.1 Three planners

For comparison of the experimental results, we considered three types of planners.

(1) The *simple* naïve planner in which the error propagated with a pre-planned fixed ϕ. This planner's anticipated uncertainty is greatly increased over the other two planners.

(2) The *adjustable* planner updates the ϕ parameter based on the current object state. This update is in line 7 of Algorithm, 2.

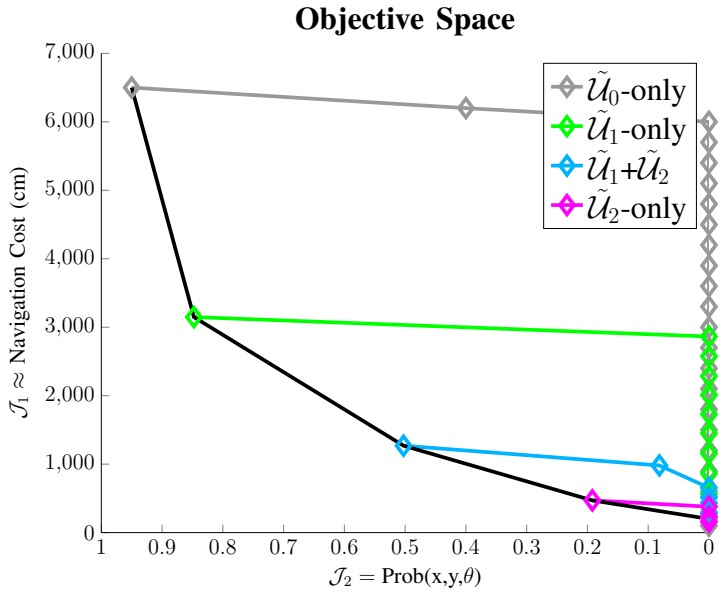
(3) The *adjustable + replanning* planner extends the previous planner by generating a new tree and a path to reduce any error that remains after the robot finishes each motion.

3.5.2.2 Two environments

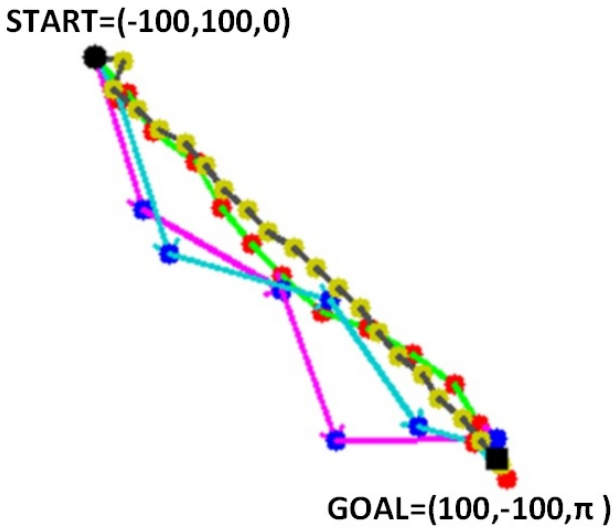
We have two scenarios. The first is obstacle-free, in which we validate the approach extensively. The second environment has a static obstacle, and gives a clear demonstration of how different cost preferences (regarding collisions) are reflected in the resulting sequence of motion primitives.

3.5.2.3 Three objectives

We are interested in total execution time, a measure of path accuracy, and a measure of path safety, denoted J_1, J_2, and J_3, respectively. Two scaling degrees-of-freedom are suffice: α_1 and α_2, with $\alpha_3 = 1 - \alpha_1 - \alpha_2$. where α_1 indicates a parameter of total execution time, α_2 is a parameter of path accuracy, and α_3 is a parameter of path safety.



(a) The Pareto front (in black); the x-axis is a measure of path accuracy; the y-axis is navigation cost.



(b) Four paths are plotted in 2-dimensional space (heading has been omitted).

Figure 3.9: $\tilde{\mathcal{U}}_0$-*only* is a gray path. $\tilde{\mathcal{U}}_1$-*only* is a green path. $\tilde{\mathcal{U}}_2$-*only* is a magenta path. $\tilde{\mathcal{U}}_1$+$\tilde{\mathcal{U}}_2$ is a sky blue path.

At transitions between motion primitives, we compute $\mathcal{J}(X)$ by accumulating each cost function. The execution time is the total robot navigation time. The total accuracy of the path includes both the reliability of object transitions between nodes, and how close the states are to the goal. They are computed via $1 - Pr(X_{oi})_{\hat{X}_{oi}}$ and $1 - Pr(X_{oi})_{X_G}$, where $\hat{X}$ is our model's prediction.

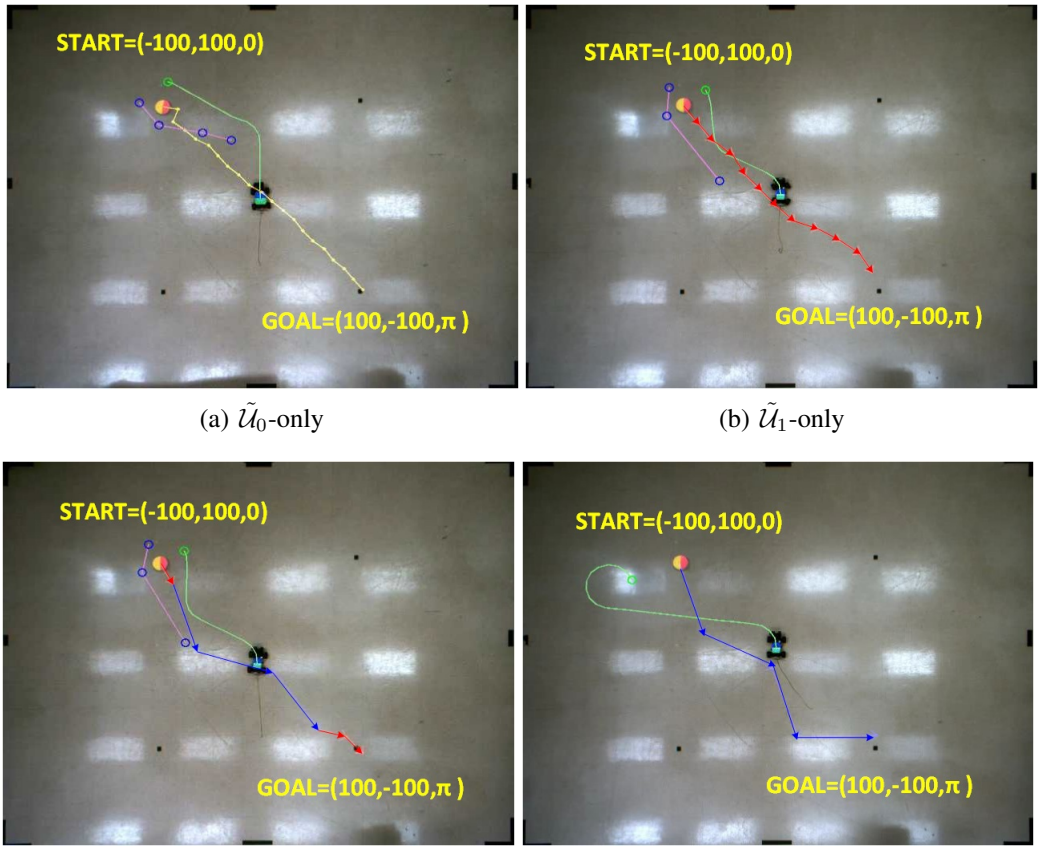
First, we examined the open space scenario to understand the basic characteristics of the motion transitions. For this scenario, we set $\alpha_3 = 0$ because there are no obstacles. Then, our four different preferences give four scaled parameters: the planner with $\alpha_1 = 0.00001$, $\tilde{\mathcal{U}}_0$-*only*, is the most conservative planner which gives a most reliable path albeit with most costly execution time. With $\alpha_1 = 0.0001$, $\tilde{\mathcal{U}}_1$-*only*, is modestly conservative, allowing a reliable path with high probability to go to the goal. The planner with $\alpha_1 = 0.0003$, $\tilde{\mathcal{U}}_2$-*only*, is most optimistic, allowing only high-speed motions. The planner with $\alpha_1 = 0.0002$, $\tilde{\mathcal{U}}_1 + \tilde{\mathcal{U}}_2$, is mixed.

As can be seen in Figure 3.9, we have four kinds of paths: (1) $\tilde{\mathcal{U}}_0$-*only* has sequences with twenty $\tilde{\mathcal{U}}_0$s. (2) $\tilde{\mathcal{U}}_1$-*only* has sequences with twelve $\tilde{\mathcal{U}}_1$s. (3) $\tilde{\mathcal{U}}_2$-*only* has sequences with four $\tilde{\mathcal{U}}_2$s. (4) $\tilde{\mathcal{U}}_1 + \tilde{\mathcal{U}}_2$ has sequences with one $\tilde{\mathcal{U}}_1$, three $\tilde{\mathcal{U}}_2$s, and two $\tilde{\mathcal{U}}_1$s.

3.5.3 Experimental Validation

Two measurements are used for comparison: *precision* measures how consistent results are across repetitions; *accuracy* considers closeness of the mean of a set of measurements to the actual goal.

Our tree was built so that the proportion of $\tilde{\mathcal{U}}_0$, $\tilde{\mathcal{U}}_1$ and $\tilde{\mathcal{U}}_2$ were equally distributed over 250,000 nodes. The start location is $X_S = (-100\ cm, 100\ cm, 0\ rad)$ and the goal location is $X_G = (100\ cm, -100\ cm, \pi\ rad)$. To simplify our experiments, we assume that the object rotates in a clockwise direction, so all motion primitives have a fixed direction of rotation.



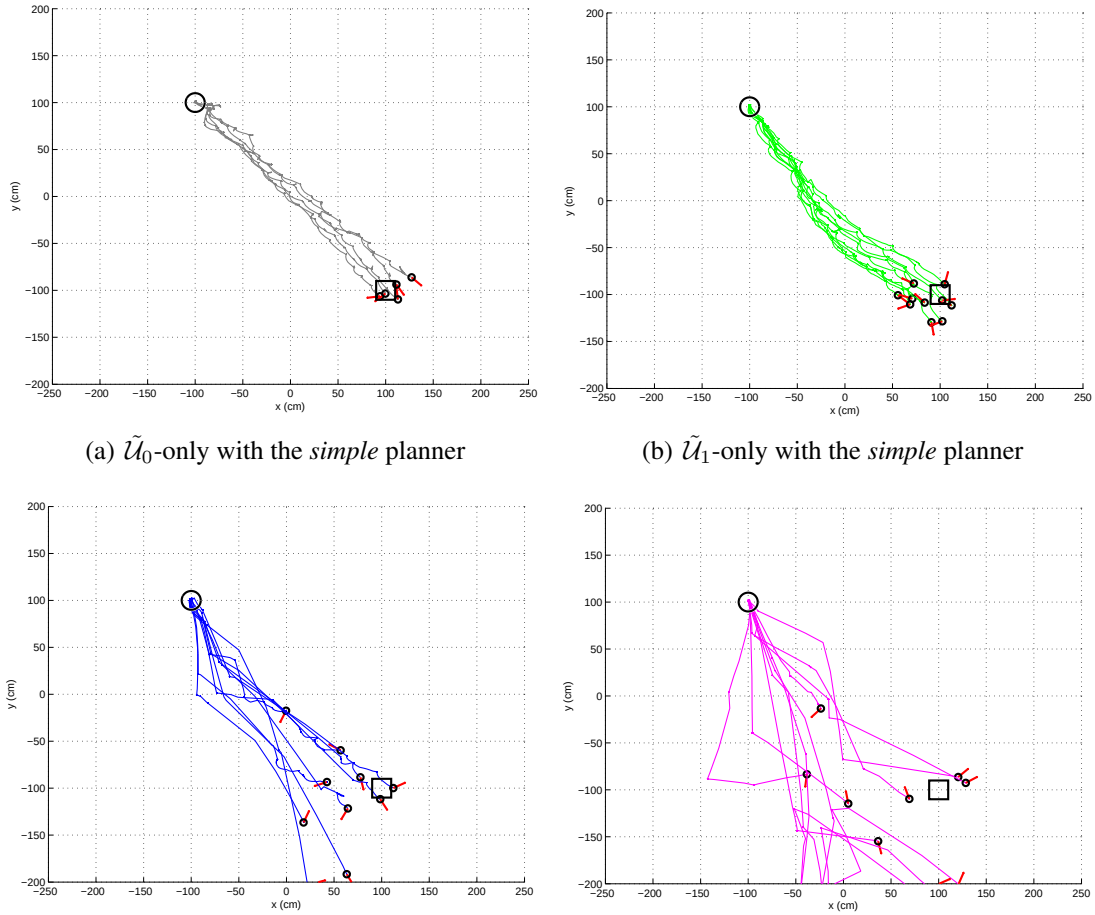
(a) $\tilde{\mathcal{U}}_0$-only (b) $\tilde{\mathcal{U}}_1$-only

(c) $\tilde{\mathcal{U}}_1 + \tilde{\mathcal{U}}_2$ (d) $\tilde{\mathcal{U}}_2$-only

Figure 3.10: An overview of our four scenarios: The small circles near the goal location indicate the final object states (orientation as a red line).

Figure 3.10 – 3.13 shows the three different planners *simple*, *adjustable*, and *adjustable + replanning* with four preferences. Figure 3.10 shows four scenarios by plotting paths (Figure 3.9) in the real world. This shows the initial state, and we can see a representative initialization motion (in green) for each primitive. Ten trials are shown for each case as follows: Figure 3.11 shows the results with *simple* planner of $\tilde{\mathcal{U}}_0$*-only*, $\tilde{\mathcal{U}}_1$*-only*, $\tilde{\mathcal{U}}_1$+$\tilde{\mathcal{U}}_2$, and $\tilde{\mathcal{U}}_2$*-only*. Figure 3.12 shows the results with *adjustable* planner analogously.

We can see that the *adjustable* planner has increased precision compared to the *simple* planner. From $\tilde{\mathcal{U}}_0$*-only* to $\tilde{\mathcal{U}}_2$*-only*, we see that accuracy also decreases. From $\tilde{\mathcal{U}}_2$*-only* of



(a) $\tilde{\mathcal{U}}_0$-only with the *simple* planner

(b) $\tilde{\mathcal{U}}_1$-only with the *simple* planner

(c) $\tilde{\mathcal{U}}_1 + \tilde{\mathcal{U}}_2$ with the *simple* planner

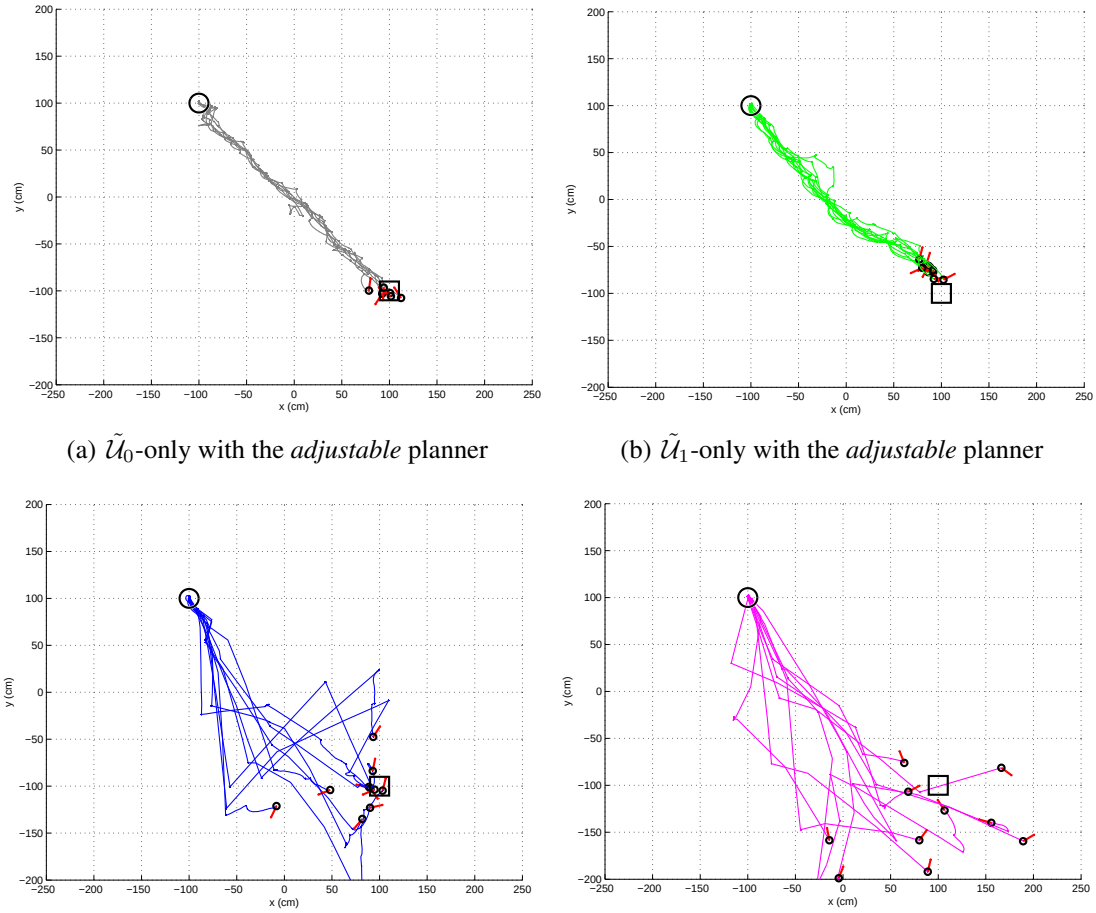
(d) $\tilde{\mathcal{U}}_2$-only with the *simple* planner

Figure 3.11: An overview of our results with the *simple* planner: The small circles near the goal location indicate the final object states (orientation as a red line).

Figure 3.11 and Figure 3.12, we might think that $\tilde{\mathcal{U}}_2$ alone is useless. However, those preferences can be effective with the *adjustable + replanning* planner; applying the *adjustable + replanning* planner for $\tilde{\mathcal{U}}_1 + \tilde{\mathcal{U}}_2$ gives the (representative) result magnified in Figure 3.13.

The replanning phase greatly increases accuracy. The replanning phase takes considerable computational time but certainly improves the quality of both $\tilde{\mathcal{U}}_2$*-only* and $\tilde{\mathcal{U}}_1 + \tilde{\mathcal{U}}_2$.

A detailed analysis of all the results appears in Figure 3.14. Figure 3.14(a) shows the model's prediction (blue), the *simple* planner (green), the *adjustable* planner (red), and



(a) $\tilde{\mathcal{U}}_0$-only with the *adjustable* planner

(b) $\tilde{\mathcal{U}}_1$-only with the *adjustable* planner

(c) $\tilde{\mathcal{U}}_1 + \tilde{\mathcal{U}}_2$ with *adjustable* planner

(d) $\tilde{\mathcal{U}}_2$-only with the *adjustable* planner

Figure 3.12: An overview of our results with the *adjustable* planner: The small circles near the goal location indicate the final object states (orientation as a red line).

the *replanning* planner (magenta). The models consistently underestimate execution cost, likely due to difficulties in controlling the RC car robustly. The estimate's deficiencies were traced back to costs in the initialization steps for the primitives.When errors are big enough, compared to given trajectories, re-planning the robot path takes additional time. Primitive $\tilde{\mathcal{U}}_0$ especially has a substantial gap between the model and reality, owing to the reversing motions further exacerbated by a sequence of more than twenty $\tilde{\mathcal{U}}_0$s.

From Figure 3.14(b), adjustment and/or replanning reduce distance errors. The bar

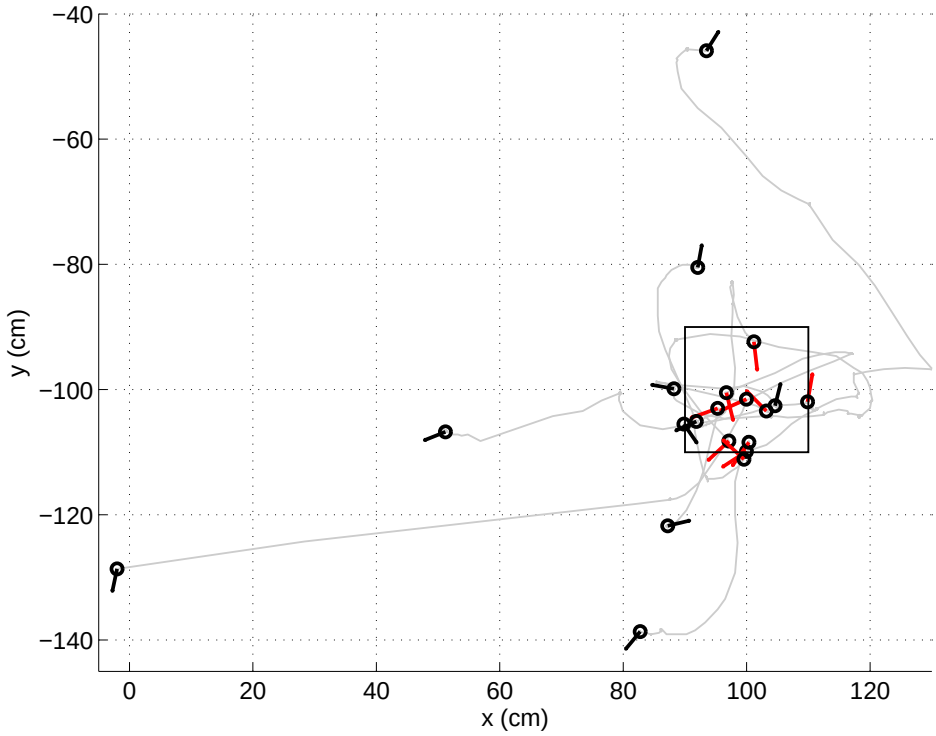
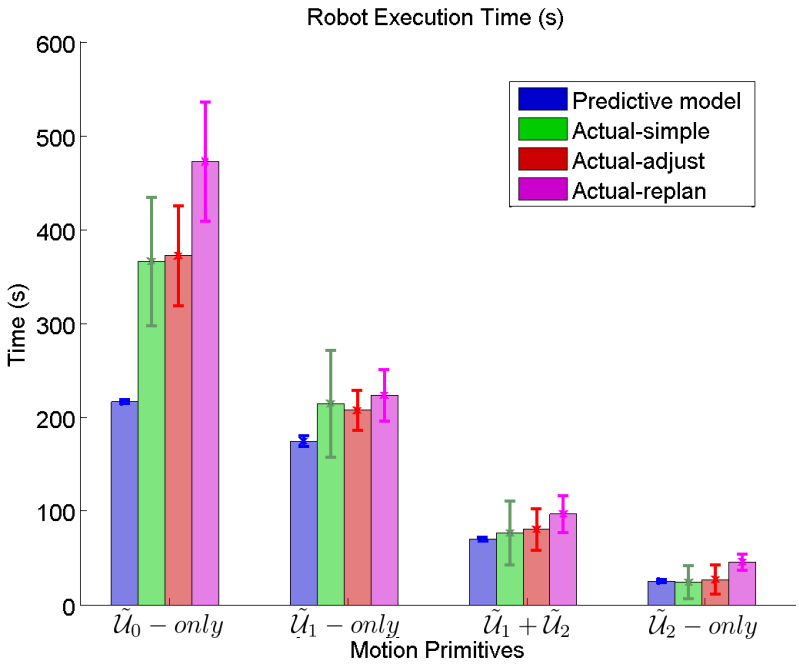


Figure 3.13: We apply the replanning phase after the result of Figure 3.12(c). Ten scattered objects are moved to the near goal location with good accuracy and precision.

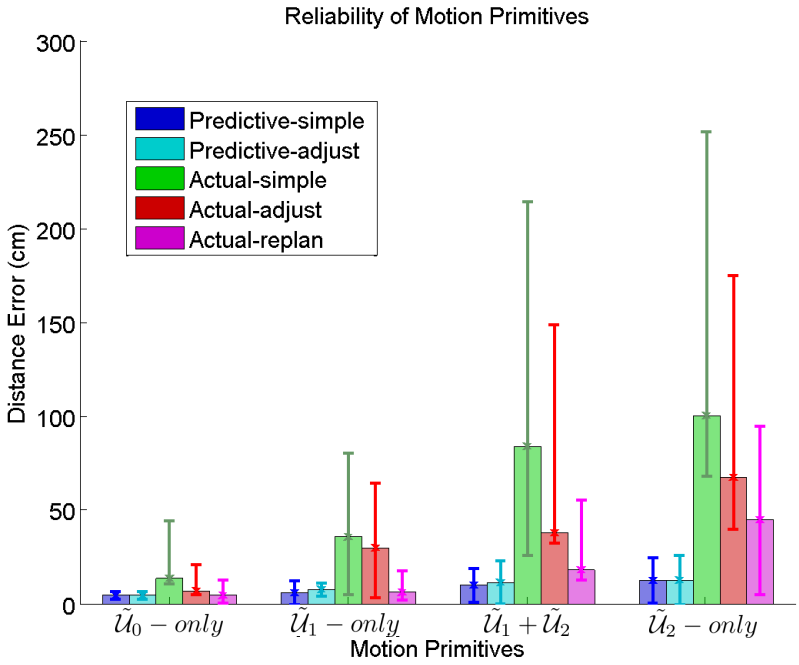
shows the mean square error, the error-bars depict the maximum and the minimum values.

The additional execution time for replanning in Figure 3.14(a) comes from the execution time of additional motion primitives. Note that the times reported for replanning do not consider the time necessary for generating the tree, as they only reflect execution costs not computational ones.

Next, we briefly summarize results for an exploration of cost preferences in the environment with obstacle collisions. In this case, we use α_2 for the scaled cost function. We have three preferences, each shown in Figure 3.15. Path-1 ($(\alpha_1, \alpha_3) = (0.0003, 0.1)$) consists of three $\tilde{\mathcal{U}}_2$s and two $\tilde{\mathcal{U}}_1$s. Path-2 ($(\alpha_1, \alpha_3) = (0.0003, 0.4)$) has four $\tilde{\mathcal{U}}_2$s and three

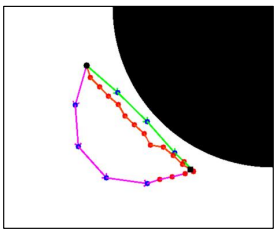


(a) Primitive execution times (mean and standard deviation).



(b) Error as distance from goal (mean, min, and max).

Figure 3.14: The data in Figure 3.11 and Figure 3.12 summarized.



Motion Primitive	Arrival time (s)	Collision rate	Arrival rate
Path 1 (green)	(84.2, 9.17)	40%	60%
Path 2 (purple)	(110, 10)	0%	80%
Path 3 (red)	(360, 15)	0%	100%

Figure 3.15: Analysis of the preferences in the obstacle environment. The black area is the static obstacle area. Reported arrival time (seconds) is (μ, σ) for 10 trials of each path. We count the number of collisions. We also count the number of arrivals at the destination ($\pm 30\ cm$ radius).

$\tilde{\mathcal{U}}_1$s. Path-3 ($(\alpha_1, \alpha_3) = (0.0001, 0.1)$) has twelve $\tilde{\mathcal{U}}_1$s. Values are reported for 10 trials for each path. Path-1 is the fastest method, but it highest risk of collision (at $40\ \%$). Path-2 has a reasonable execution time and no collision. Path-3 is the most conservative path.

	Fine dragging $\tilde{\mathcal{U}}_0$	Appr. dragging $\tilde{\mathcal{U}}_1$	Conjoining + [Bhattacharya et al., 2015]	Hooking	Snare + dragging	Striking $\tilde{\mathcal{U}}_2$	Hooking + Striking
Properties	Statics-based idealized & interaction	Statics-based stick/slip together	Magnets included	Either Statics or dynamics-based	Statics-based stick/slip together	Dynamics-based	Dynamics-based
Model	Deterministic w/ small variance added	Stochastic	Simulation-based	Simulation-based	Data-driven (Stochastic)	Data-driven (Stochastic)	Simulation-based or Data-driven (Stochastic)

Figure 3.16: We extended a small portfolio in Figure 3.5 to a diverse set of motion primitives.

3.6 Extension of a Set of Motion Primitive

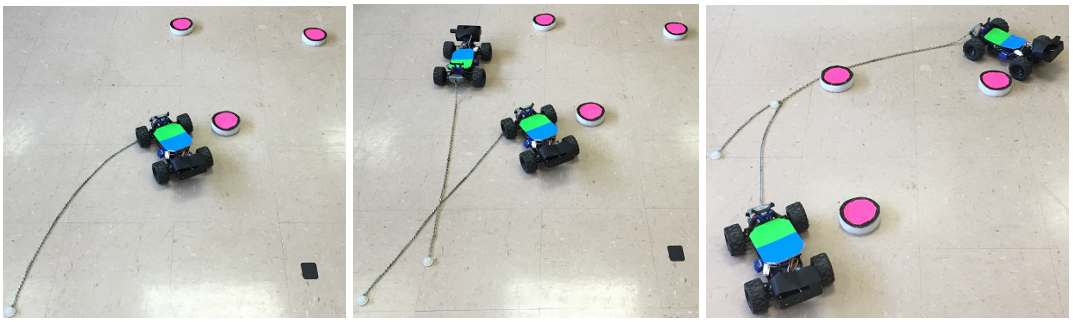
We reviewed a small set of motion primitives in Figure 3.5. In this section, we show a diverse set of motion primitives as shown in Figure 3.16. We explain additional motion primitives: *conjoining*, *hooking*, and *snaring*.

3.6.1 A Conjoining Motion Primitive

We developed a docking primitive that is necessary to conjoin two robots as a pair of robots. Firstly, given objects to move together, one robot stays near one object, initializing the robot's orientation to transfer (Figure 3.17(a)), then the second robot crosses the tail of the first (Figure 3.17(b)). Finally, the tails joined as the robot follows the contour of the convex hull because each tail contains a magnet. See Figures 3.17(b) and 3.17(c). After that, a pair of robots moves clustered objects to goal locations. To split conjoined tails, we simply make a extra motion: one robot stops while the other moves past until the length of the tails is exceeded.

3.6.2 A Hooking Motion Primitive

We devised this motion because this motion facilitates caging effectively objects and moving to a goal location by one robot. To produce this hooking primitive, we put a fist-



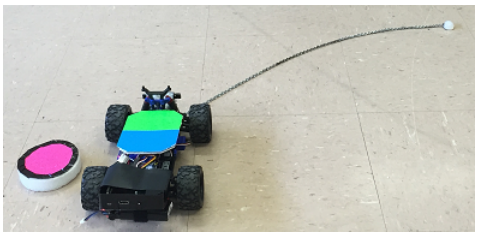
(a) Docking: T = 0 sec. (b) Docking: T = 8 sec. (c) Docking: T = 12 sec.

Figure 3.17: Initializing docking in (a). One robot stays at the given location. The second robot crosses the tail of the first. Since each tail, made of chain, has a magnet at the end, the two tails join naturally so long as the robot follows the contour of the convex hull via (b) to (c).

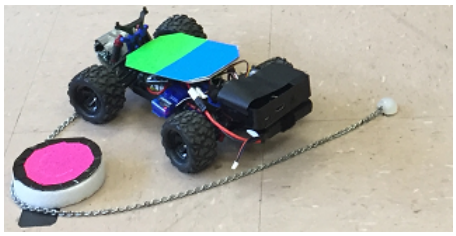
size quarter sphere at the end of the tail (see Figure 3.18(a)). First, the robot winds around the object, crossing near the end of its tail (Figure 3.18(b)). Then, the quarter sphere works like a buckle (Figure 3.18(c)). Thereafter, the robot moves the hooked object to a goal location. To release hooked objects, the robot turns around to the object reversing the direction of the winding motions (Figure 3.18(d) to Figure 3.18(f)).

3.6.3 A Snaring Motion Primitive

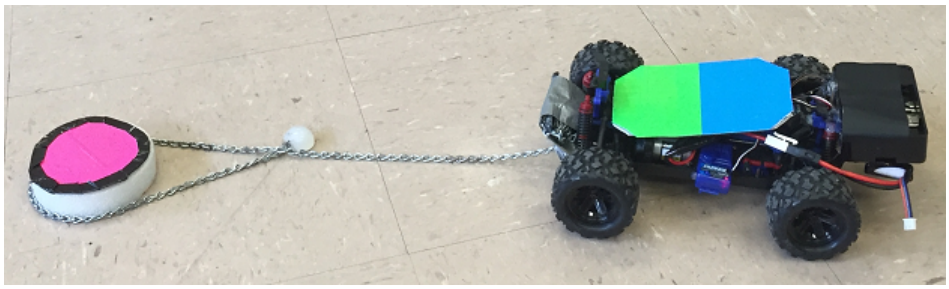
We also developed a *snaring* motion primitive that winds around the objects. In contrast to the previous motion primitive $\tilde{\mathcal{U}}_1$, this motion has one special advantage. When objects are too heavy to move with a dragging motion, the snaring motion creates more friction between the tail and an object so that the object is under control by the wrapped tail. Figure 3.19(a) shows the robot winding the object for 5 times. By the end, the robot can drag the gripped object into the next desired states with unwrapping the tail. Figure 3.19(b) shows the robot going to the desired orientation, then wrapped object follows the robot's trajectory by unwinding the tail. In this motion primitive, we might be able to use a simple stochastic model same as a striking motion, $\tilde{\mathcal{U}}_2$.



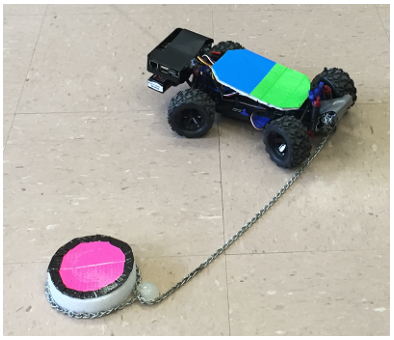
(a) Hooking: T = 0 sec.



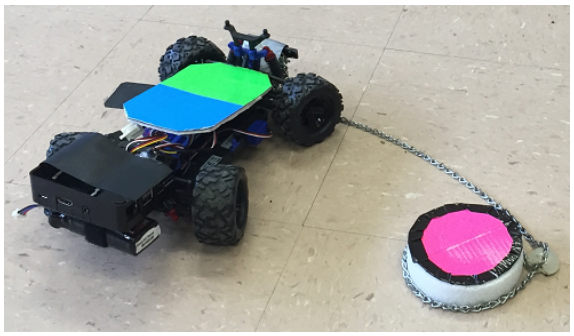
(b) Hooking: T = 5 sec.



(c) Hooking: T = 9 sec.



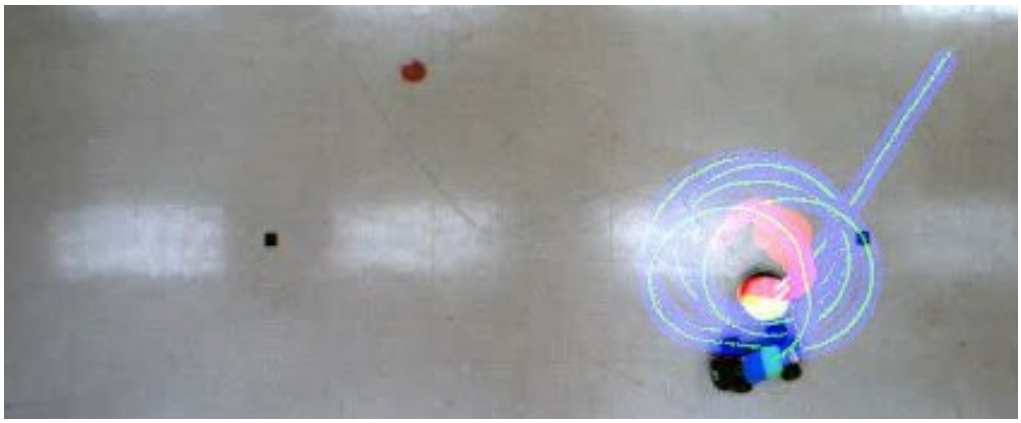
(d) Unhooking: T = 0 sec.



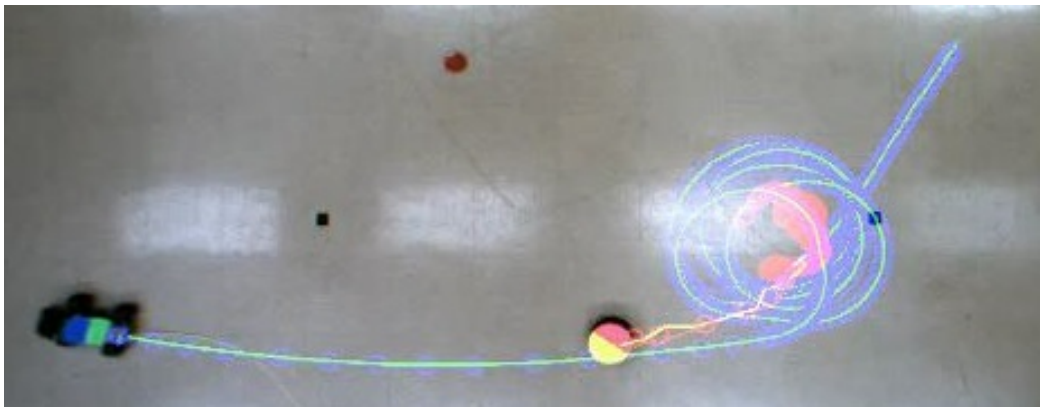
(e) Unhooking: T = 3 sec.

(f) Unhooking: T = 12 sec.

Figure 3.18: The robot winds around the object with a clockwise direction through (a), (b) and (c). To release the hooked object, the robot drives in the opposite direction, seen in (d), (e) and, (f).



(a) The robot winds around the objects for 5 times, and then keeps surrounding the object to find the orientation to go.



(b) The robot goes to the desired orientation. Then, the object follows the robot's trajectory by unwinding the tail.

Figure 3.19: We developed a snaring motion primitive, which is useful when the object is heavy to move by a dragging motion. The object can be under control by this snaring and dragging motion.

We will employ the hooking motion primitive and the conjoining motion primitive in Chapter 4.

3.7 Discussion

3.7.1 Implementation Challenges

An issue we encountered with our implementation was that it was difficult to assure that the robot maintained a fixed velocity. This is because friction varies depending on whether the object (and tail, including just some segments of it) are moving or not. A consequence is that the total execution time reported above is computed based on the average velocity of the robot.

3.7.2 Object Geometry Oblivious: a Generalization of our Model

When we use a rope-like structure, it can wrap around an object, conforming to any object geometry due to the flexibility of the tail. Thus we expect that the same wrapping/striking model will work for many geometries.

To demonstrate the usefulness of a rope-like structure, we studied with several different object shapes using the same model in our proposed work. Based on our experiments, we conclude that our approach is not only limited to cylindrical objects. If the shapes have the comparable enclosing length via the rope-like structure (i.e., perimeter), and approximate locations of center of mass and center of friction, then they appear to work well with the same parameterized models.

Moreover, this is not limited only to convex polygons; concave polygon such as a star shapes work because they can be treated effectively as their convex hulls when the rope-like structure surrounds them.

We explored four different objects (*e.g.,* a circle, a square, a triangle, and a star-shape). The robot and tail started in some randomly generated initial configurations, then we executed two primitives, $\tilde{\mathcal{U}}_1$ and $\tilde{\mathcal{U}}_2$ at least 10 times to collect sufficient data. The initial pose of the object is $(0, 0, 0)$. The dots in the Figure 3.20 indicate the final object position (the orientation is not displayed here, but listed in Table 3.1). Figure 3.20(a) shows four

different shapes of the object executed by $\tilde{\mathcal{U}}_1$, while Figure 3.20(b) is corresponds to the $\tilde{\mathcal{U}}_2$ execution. Table 3.1 shows an analysis (mean and standard deviation) of the collected data.

For $\tilde{\mathcal{U}}_2$, Figure 3.20(a) shows high accuracy and high precision results with regard to the different shapes of the object. Figure 3.20(b) also shows high precision results for each shape, but lower accuracy. The reason is that when the tail struck the object, the orientation is less controllable for $\tilde{\mathcal{U}}_2$. However, the object's total distance moved shows great consistency for all different shapes of the objects.

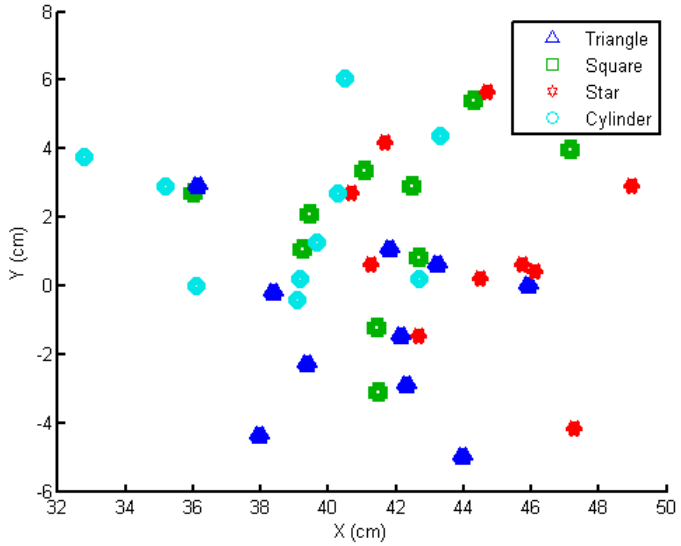
In conclusion, with the traditional approaches to non-prehensile manipulation [26, 28, 29, 108], the robot manipulates an object through a direct contact; in this case, depending on the shape of the object, manipulating an object might be intractable. However, if the robot with a compliant, unactuated rope-like structure surrounds an object in some particular ways (via dragging or striking), then the shapes of the object matters comparatively little in the same model. We might call this an *object geometry oblivious* approach, suggesting the generalizability of our model.

3.7.3 The Advantage of the Dynamics-based Motion Primitives

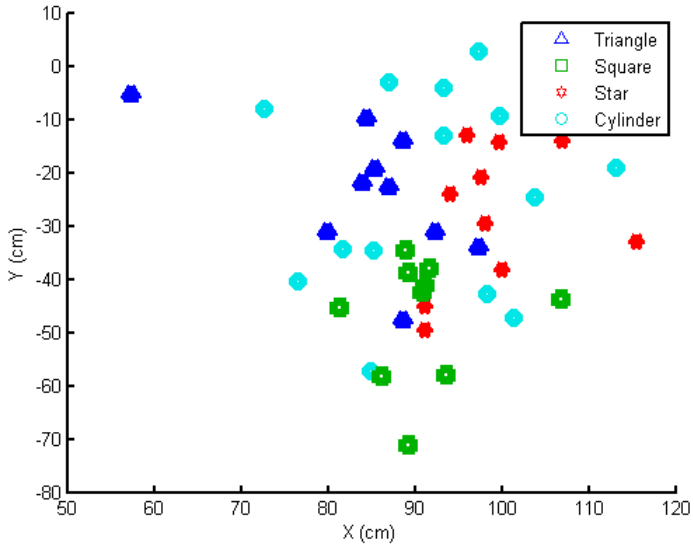
We present some practical particular scenarios that show the advantage of the dynamics-based motions. We mentioned that one of our contributions is employing the dynamics of

	$\tilde{\mathcal{U}}_1 = (cm, cm, rad)$	$\tilde{\mathcal{U}}_2 = (cm, cm, rad)$
Triangle	$(41.1, -1.2, 6.6) \pm (3.1, 2.5, 0.6)$	$(84.5, -23.8, 4.5) \pm (10.7, 12.7, 0.9)$
Square	$(41.5, 1.8, 6.4) \pm (3.0, 2.5, 0.5)$	$(90.9, -47.2, 4.2) \pm (6.5, 11.5, 0.6)$
Star	$(44.4, 1.2, 7.0) \pm (2.7, 2.8, 0.5)$	$(99.1, -28.2, 4.3) \pm (7.4, 13.2, 0.9)$
Cylinder	$(38.8, 2.1, 6.5) \pm (3.3, 2.2, 0.6)$	$(92, -23.9, 3.6) \pm (11.3, 18.9, 1.0)$

Table 3.1: We used a chain with 35 g and 70 cm. The mass of each object is 23 $g \pm 2$ g.

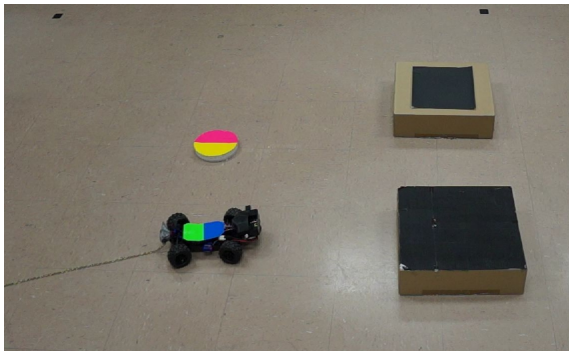


(a) A dragging primitive $\tilde{\mathcal{U}}_1$ with four different types of objects

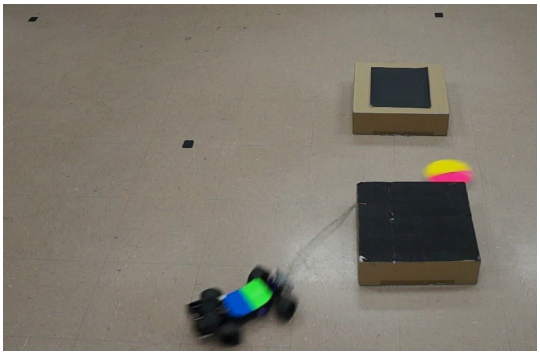


(b) A striking primitive $\tilde{\mathcal{U}}_2$ with four different types of objects

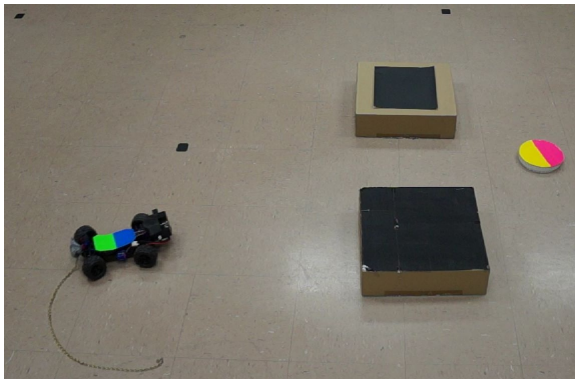
Figure 3.20: Experiments to determine whether the shapes of the object are critical for our system. The initial object pose is $(0, 0, 0)$.



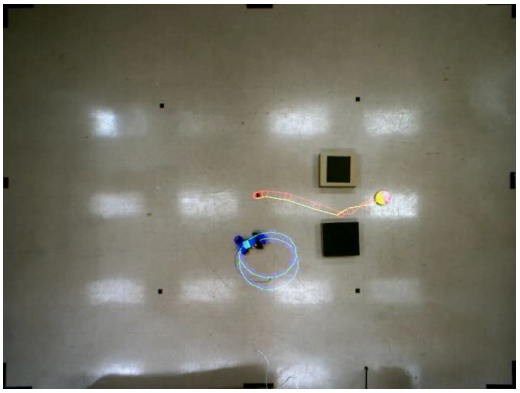
(a) Penetrating a narrow passage: T = 0 sec.



(b) Penetrating a narrow passage: T = 0.8 sec.



(c) Penetrating a narrow passage: T = 1.2 sec.



(d) Penetrating a narrow passage: tracking by an overhead camera

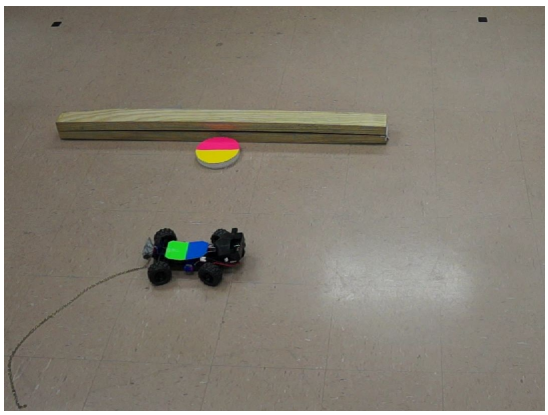
Figure 3.21: Using $\tilde{\mathcal{U}}_2$, the snapshots from (a) to (c) show how an object can be moved through a narrow passage even though the robot is too wide to pass through it (*e.g.,* pushing a paper under the door); (d) shows an incremental tracking information.

rope-like structures for manipulating objects. Here we demonstrate practical examples for $\tilde{\mathcal{U}}_2$ (a striking primitive).

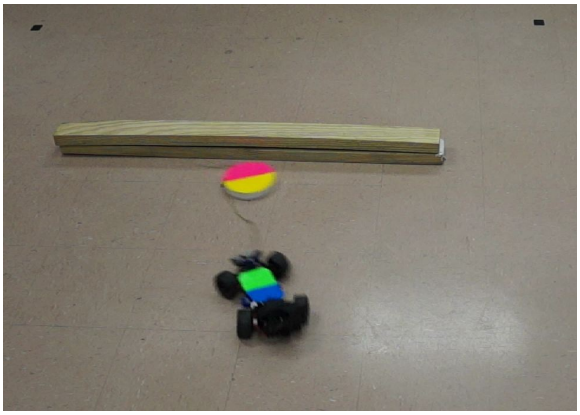
The first scenario is that we manipulate an object to the goal location through a particular environment such as a corridor shown in Figure 3.21. In this scenario it is difficult to manipulate an object via non-prehensile manipulation. For example, assuming the robot cannot pass through the narrow passage due to its size. Then, by using $\tilde{\mathcal{U}}_2$, we can generate a high-speed striking motion that can move an object quickly and reliably through the

passage. This narrow passage can be treated as obstacles (like a static obstacle environment in Section 3.5.3), and then we can easily generate a path using our framework in this section. Figure 3.21 shows an object can be moved via a narrow passage using the striking primitive, $\tilde{\mathcal{U}}_2$.

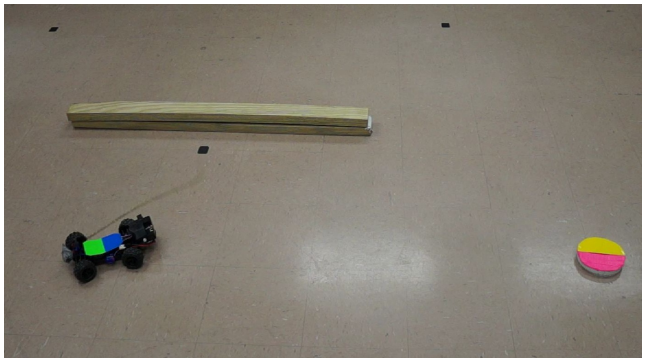
Second and third scenarios are: the robot uses $\tilde{\mathcal{U}}_2$ for moving an object that is stuck near the obstacles. In non-prehensile manipulation, manipulating a stuck object is a challenging problem. Figures 3.22 and 3.23 show a stuck object can be whipped away from the obstacles using $\tilde{\mathcal{U}}_2$.



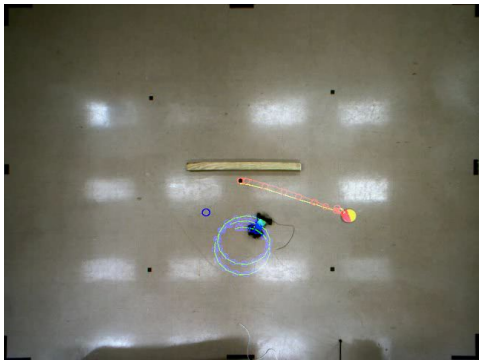
(a) A stuck object near the wall: T = 0 sec.



(b) A stuck object near the wall: T = 0.5 sec.

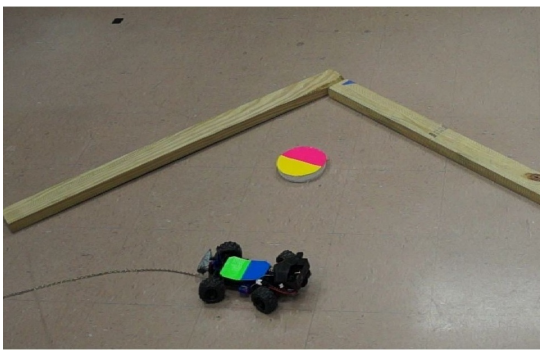


(c) A stuck object near the wall: T = 1 sec.

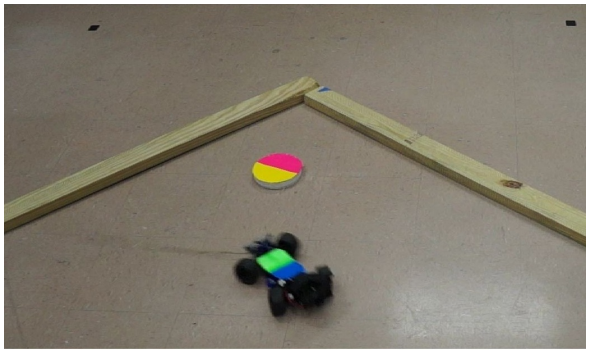


(d) A stuck object is whipped away from the wall: tracking all is by an overhead camera.

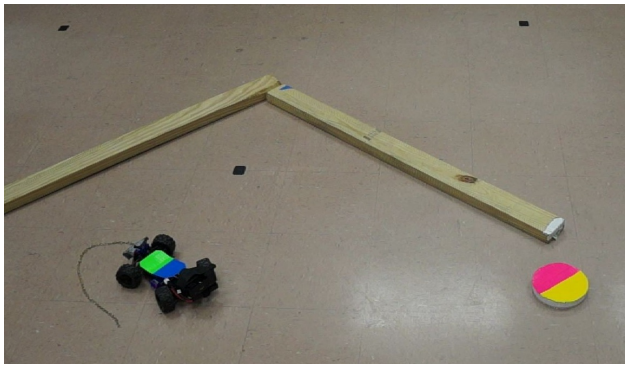
Figure 3.22: Using $\tilde{\mathcal{U}}_2$, the snapshots from (a) to (c) show how an object can be moved outwards from the wall; (d) shows an incremental tracking information.



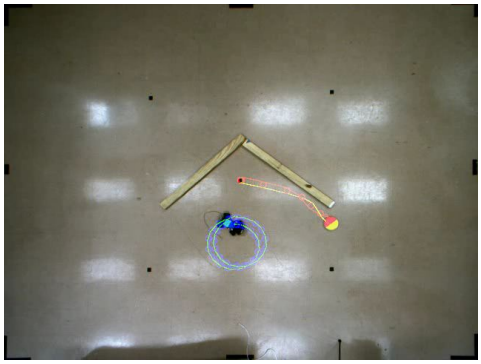
(a) A stuck object near the corner: T = 0 sec.



(b) A stuck object near the corner: T = 0.5 sec.



(c) A stuck object near the corner: T = 1 sec.

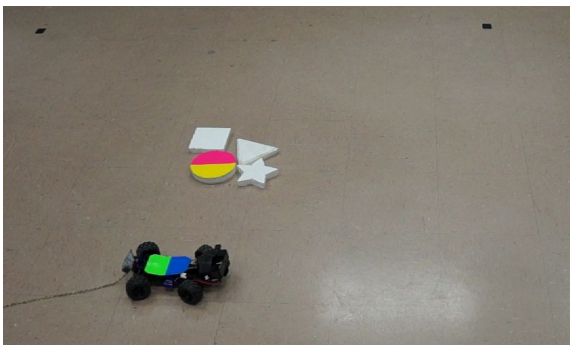


(d) A stuck object is whipped away from the corner: tracking all is by an overhead camera.

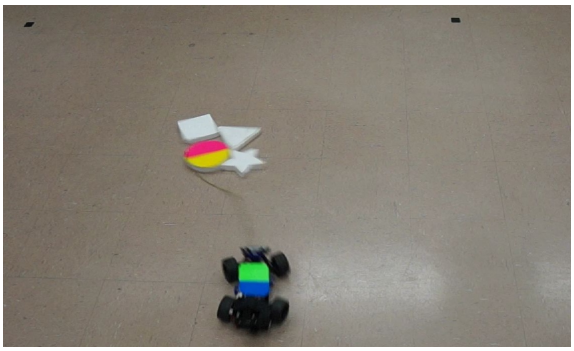
Figure 3.23: Using $\tilde{\mathcal{U}}_2$, the snapshots from (a) to (c) show how an object can be moved outwards from the wall; (d) shows an incremental tracking information.

The last scenario is that the robot with a tail uses $\tilde{\mathcal{U}}_2$ for splitting apart objects gathered in a cluster, simultaneously, so that the robot can manipulate the right object into a desired goal location. For example, when the robot wants to move gathered objects to a goal, we might need to scatter stuck objects beforehand. Figure 3.24 shows how the robot with a tail scatters stuck objects.

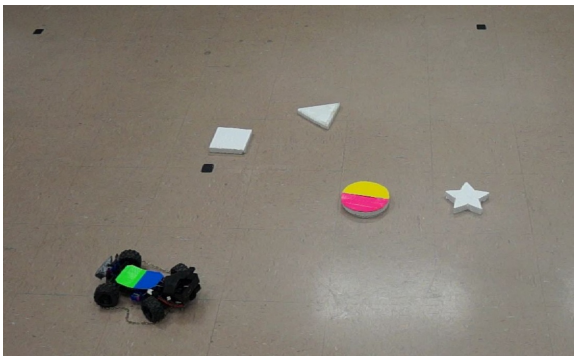
Overall, dynamics-based motions, such as $\tilde{\mathcal{U}}_2$, are not merely a scientific curiosity, they



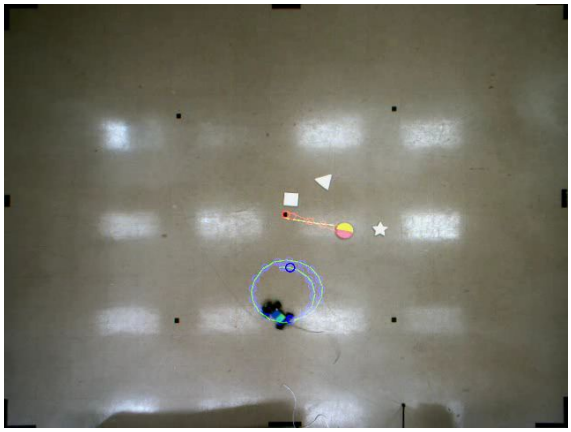
(a) stuck objects together: T = 0 sec.



(b) stuck objects together: T = 0.5 sec.



(c) stuck objects together: T = 1 sec.



(d) stuck objects together: tracking by an overhead camera

Figure 3.24: Using $\tilde{\mathcal{U}}_2$, the snapshots from (a) to (c) show how stuck objects can be scattered; (d) shows an incremental tracking information. Here we only tracked the cylindrical shaped object.

can be useful in several practical scenarios. When a robot cannot access a narrow passage, a high-speed striking motion can move an object quickly and reliably through the passage. If an object is stuck near a wall or a corner, the robot can use $\tilde{\mathcal{U}}_2$ to manipulate a stuck object directing it outwards from afar. It is also possible for a robot with a tail to use $\tilde{\mathcal{U}}_2$ to separate multiple gathered objects, so that each object can be brought into their desired location, much like a billiards break.

3.8 Summery of This Section

Even though rope-like structures are simple, cheap, and versatile tools, compliant passive tails have received little attention for robotic manipulation. We speculate that this is a consequence of the modeling difficulties inherent to such systems; this chapter's response to the challenge is to use a collection of models of varying verisimilitude. We have demonstrated how the dynamics of a compliant, unactuated tail can be successfully exploited for manipulation. We have shown the effectiveness of an approach that uses simple stochastic models, each associated with a structured primitive. The planning algorithms used was sampling-based motion planning with a particle-based representation for error propagation. We demonstrated our system with physical robot experiments and the results show that various preferences can be expressed effectively, ultimately being reflected in different mixes of primitives being executed.

4. COOPERATIVE MANIPULATION OF OBJECTS VIA COMPLIANT, UNACTUATED TAILS

4.1 Introduction

Manipulation problems that involve transferring multiple objects to a set of goal locations arise in many applications and in surprisingly diverse settings. Familiar examples include collecting toys strewn across the floor, removing debris on the surface of a pond, or (more mundanely) handling materials in a warehouse. Such problems have inherent parallelism ripe for exploitation if one can move multiple objects simultaneously. In certain circumstances, such as when the environment is large, a single robot's performance may be inadequate and a collaborative multi-robot team needed to perform such collection tasks more efficiently.

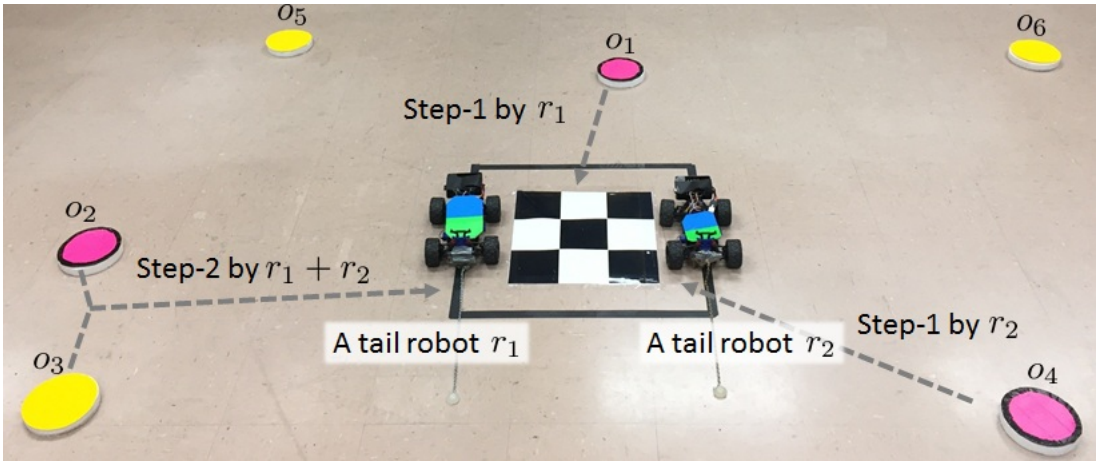
The focus of this chapter is on robots equipped with an unactuated flexible structure attached as a 'tail' that they can use as a tool. We examine how a group of robots equipped in this way can form a team to solve multi-object collection tasks. It is apparent, in reflecting on our everyday lives, that there are a huge variety of ways that people use strings, cords, ropes, lines, and chains—they are used to bind and secure, they are used to connect and pull, they are used to grip and whip. It isn't easy to find another class of artefacts more versatile! Still, most robots do not make use of ropes or strings, probably because modeling flexible structures remains challenging. But the advantages for manipulation are apparent: when wrapped to encircle a payload, ropes and strings can restrain objects through the interplay of tension and friction. Perhaps best of all, little need be known about the geometry of the objects being manipulated once the rope or string is constricted.

The present work employs two motion primitives in Figure 3.16: a hooking motion primitive and a conjoining motion primitive. We developed a robot capable of manipulat-

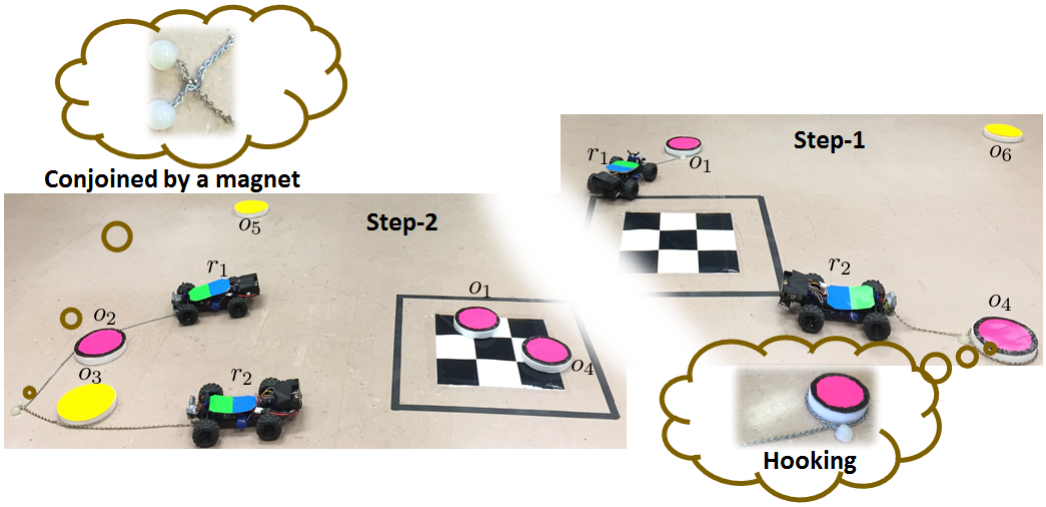
ing single objects by using the hooking motion primitive. This hooking primitive contrasts with research in which a pair of robots, connected by a rope using a conjoining motion primitive, manipulate one or more objects (*cf.* [11, 30]). Both approaches (the hooking and the conjoining primitives) are unified in this Section. Our robots can manipulate single objects individually or, if they choose, they can link their tails together (by executing a special manoeuvre) and act as a connected pair.

As an illustrative example, Figure 4.1 provides a small instance of the type of manipulation problem we tackle. The top image (Figure 4.1(a)) shows the initial state of the environment and a plan that the robots can use to solve the problem. In this case, execution of this plan involves two steps. The robots begin by manipulating objects individually by securing their tails around the objects (upper-right in Figure 4.1(b)). Once the objects have been delivered to the goal region and released, the robots connect their tails and cooperate in dragging multiple objects to the goal (lower-left in Figure 4.1(b)). When operating as a conjoined pair, we expect that robots will have greater towing capacity in concert than when acting alone. But, two robots acting individually have the advantage of being able to collect far-flung objects simultaneously.

Solutions to collection problems with these kinds of robots must resolve the question of when robots should act jointly, as tightly-knit pairs, and when they should act individually. Among other factors, the best choice depends on whether the objects are clustered within the environment, the number of robots, and their capacity when operating as a conjoined pair. Picking what we believe to be an appropriate level of abstraction, we formulate the Multi-Object Collection via Cooperative Towing Planning Problem (MOCCT) as a discrete optimal path planning problem, whose solution is a sequence of paths for all robots that minimizes the total cost of manipulation.



(a) Two robots with flexible tails construct a plan to transport four objects: Step-1 is for the robots, acting individually, to move o_1 and o_4 to the goal simultanously; Step-2 involves the pair of robots linking tails and, together, bringing o_2 and o_3 to the goal.



(b) To perform Step-1, each robot tows an object on its own, hooking its tail around the object and releasing it at the goal. In Step-2, the robots execute a primitive that physically couples the pair, they then surround and drag o_2 and o_3 to the goal cooperatively.

Figure 4.1: Consider the problem of moving four objects to the chequered region, with at least two objects being pink. To minimize cumulative distance, the robots balance working as a tightly-knit pair versus operating separately. Pairs have the advantage of greater towing capacity, while individuals may fetch distant objects concurrently.

4.2 Problem Setup and Notation

Let the set $\mathcal{O} = \{o_1, o_2, \ldots, o_n\}$ represent n objects which can be transported by our m robots: $\mathcal{R} = \{r_1, r_2, \ldots, r_m\}$. We will write $\textit{pos}^t(o_i)$ to denote the position of object i at time t, (and robots, similarly); $t \in \mathbb{N}_0$ is a temporal index, starting at zero. We will use the term *world configuration* at time t, to describe the positions of all objects and robots at that instant. A set of goal locations, $\mathcal{G} = \{g_1, \ldots, g_k\}$, is given. By extending *pos* in the natural way, $\textit{pos}^t(g_i) = X_i^G, \forall t$ gives g_i's (fixed) position.We also find it helpful to write the initial positions of the objects as X_i with $X_i = \textit{pos}^0(o_i)$, and robots' initial locations with $X_i^R = \textit{pos}^0(r_i)$. Finally, we collect all these initial positions into the set $\mathcal{X} = \{X_1, \ldots, X_n, X_1^R, \ldots, X_m^R, X_1^G, \ldots, X_k^G\}$. To describe the mass of each object, we will write the function $\textit{mass}(o_i)$ to denote the mass for o_i, then let the set $\mathcal{D} = \{\textit{mass}(o_1), \ldots, \textit{mass}(o_n)\}$.

Along with positions, each robot's state represents that it is either operating alone or has its tail connected to another robot. Let $\textit{match}^t(r_i) = r_j$, with $i = j$ if and only if r_i is operating solo and with $i \neq j$ if and only if r_i and r_j form a couple together. We partition $\mathcal{R}$ into $\mathcal{R}_s^t \cup \mathcal{R}_p^t$, where solo robots comprise $\mathcal{R}_s^t$, pairs $\mathcal{R}_p^t$.

Further, assume two sets of logical predicates are given:

$\mathfrak{L}_{\text{static}}$ contains predicates describing static properties of the objects, like $\text{P} : \mathcal{O} \to \{\text{True}, \text{False}\}$; e.g., $\text{PINK}(\cdot)$, $\text{CYLINDER}(\cdot)$, $\text{WOOD}(\cdot)$, $\text{BOOK}(\cdot)$, $\text{BALL}(\cdot)$, etc.

$\mathfrak{L}_{\text{pos}}$ with properties defined as $\text{P}' : \mathcal{O} \times \mathcal{X} \to \{\text{True}, \text{False}\}$, that include position information; e.g., $\text{LOCATEDAT}(\cdot, \cdot)$.

A set of desired final states is specified via a logical formula, $\mathbf{F}$, written in terms of $\mathfrak{L}_{\text{static}}$ and $\mathfrak{L}_{\text{pos}}$, which, when satisfied in some world configuration, distinguishes it as a goal. Rather than providing a complete BNF characterization of the formulas, which would

only belabor the point, we observe that in addition to the standard logical connectives ($\neg$, $\wedge$, $\vee$), SMT also enables one to write relationships between the cardinalities of sets (and natural numbers) with $\leq, <, \neq$, etc. Note that the goal locations $\{X_i^G | g_i \in \mathcal{G}\}$ enter into this specification as being part of the domain of predicates in $\mathfrak{L}_{\text{pos}}$.

To describe meaning to all sentences of a first-order logic, we define the domain of discourse U for interpretations: Let the set $\mathbb{O} \triangleq \{obj_1, obj_2, \ldots, obj_n\}$ represent n objects. Let the set $\mathbb{L} \triangleq \{Loc_1, loc_2, \ldots, loc_v\}$ represent v locations. We define interpretation $\mathcal{I}$ at time t, which maps constant symbols, predicate symbols to relations on the domain of discourse as follows:

$$\mathcal{I}^t = \begin{cases} obj_i \in \mathbb{O} \mapsto o_i \in \mathcal{O} \\ Loc_i \in \mathbb{L} \mapsto X_i \in \mathcal{X} \\ \text{LOCATEDAT} \mapsto \{\langle o_i, X_k \rangle \mid o_i \in \mathcal{O}, X_k \in \mathcal{X},\ pos^t(o_i) = X_k\} \\ etc. \end{cases} \tag{4.1}$$

We might be able to express complicated relations by using the firs-of-logic: We can include sets, lists, natural numbers (by the Peano Axioms). We observe that in addition to the standard logical connectives ($\neg$, $\wedge$, $\vee$), SMT also enables one to write relationships between the cardinalities of sets (and natural numbers) with $\leq, <, \neq$, etc. Thus we can design general constraints and solve it by using SMT-solver.

4.2.1 The MOCCT *Problem Formulation*

We formulate the MOCCT problem as a task planning problem using high-level motion primitives [85] for the robots' movements. The sequence of primitives comprises a motion plan having semantics defined in terms of a C-space motion (details appear later).

In this chapter, manipulation proceeds, whether by a single robot or conjoined pairs,

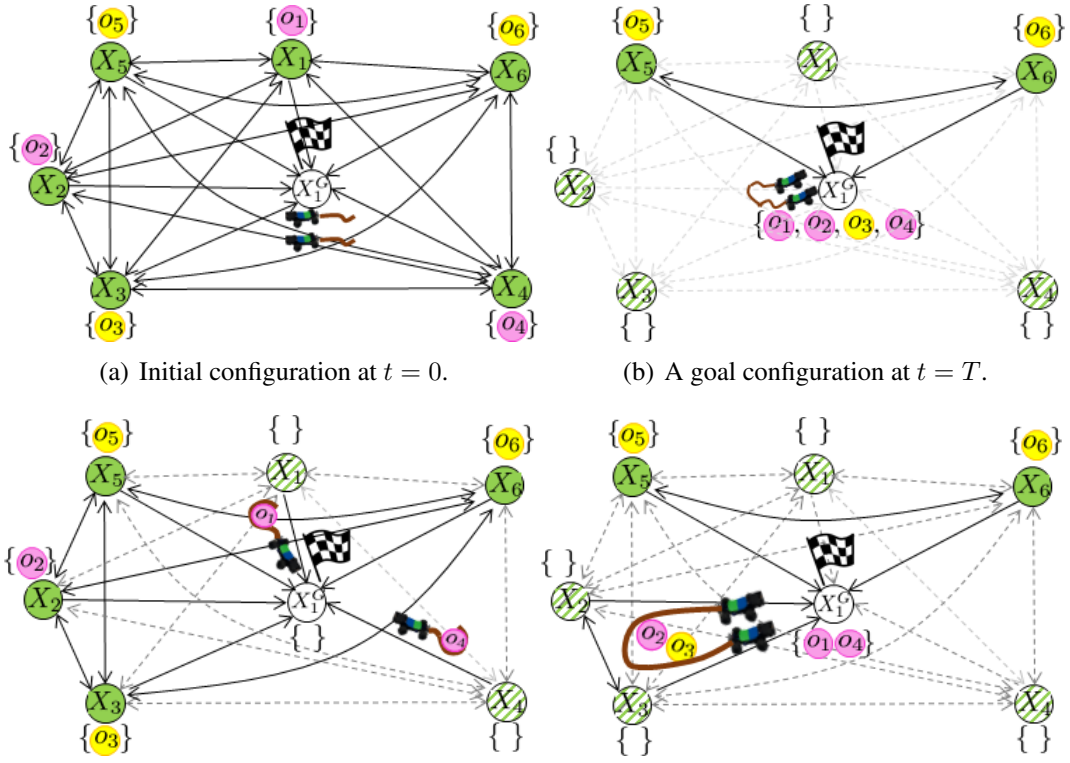
by encircling a payload and then towing it to some position. Once multiple objects are brought together, the robots are incapable of separating them again.

Modeling assumption: We impose the natural restriction that all the objects at each $X \in \mathcal{X}$ must be moved together. It is useful to have notation for the set of all objects at some location, with the mnemonic for a bundle, we write $b_X^t = \{o_i : \mathcal{O} \mid pos^t(o_i) = X\}$, where $X \in \mathcal{X}$; the X for b_X^t is the bundle's site. One consequence of the preceding modeling assumption is that we do not consider plans of sequences longer than the total number of objects; we denote this maximum $\bar{T}$. Another is that we can talk meaningfully about multiple objects at $X \in \mathcal{X}$, because we treat them as indivisible thereafter so their centroid suffices to characterize the state that is important to us.

The search space over world configurations is now clear: each element of the search space consists of a set of bundles, their associated sites, and the robots' state. We define planning operators which transition one world configuration to another. It is easiest to understand the world configurations and the valid transitions between them by visual means. Fig. 4.2 illustrates the scenario of Fig. 4.1(a). Bundles of objects appear in braces at each location. If, at time t, elements from site X_i are moved to X_j, the result of the operation will be $b_{X_j}^{t+1} = b_{X_i}^t \cup b_{X_j}^t$. The figure abstracts away several details, but these are clarified by looking at the planning operators in terms of the motion primitives the robots use to mediate their operation in the environment. In the following four primitives, we use $\mathbf{r}$ to denote a set of robots, either a singleton $\mathbf{r} = \{r_k\}$, $r_k \in \mathcal{R}_s^t$, or a pair $\mathbf{r} = \{r_k, r_\ell\}$, $\mathbf{r} \subseteq \mathcal{R}_p^t$.

1. TRANSIT$(\mathbf{r}, X_i)$: This primitive returns a path Δ_1 for moving robot(s) in $\mathbf{r}$ to X_i without colliding with obstacles and objects. It changes the robot(s) locations to X_i, initializing their orientation so they are ready to execute TRANSFER$(\cdot)$.

2. TRANSFER$(\mathbf{r}, X_i, X_j)$: This primitive returns a path Δ_2 for collision-free motion of robot(s) in $\mathbf{r}$ so that all $o_k \in b_{X_i}^t$ arrive at X_j. This changes the locations of both robot(s)



(a) Initial configuration at $t = 0$.

(b) A goal configuration at $t = T$.

(c) Step-1: two solo robots transfer o_1 and o_4 to the goal concurrently.

(d) Step-2: a pair transfer o_2 and o_3 to the goal working together.

Figure 4.2: A directed graph representation for the example in Fig. 4.1(a). A green circle indicates a node X_i and a curly bracket shows a set of objects $b^t_{X_i}$. Here (a) represents an initial configuration while (b) is a goal configuration; (c) shows Step-1 in Fig. 4.1(a), while (d) shows Step-2 in Fig. 4.1(b). Heuristics mean some edges (dotted) are unlikely to be explored.

and object(s) to X_j.

3. DOCK$(\mathbf{r}, X_i)$: For $\mathbf{r} = \{r_k, r_\ell\}$, $\mathbf{r} \subseteq \mathcal{R}^t_s$, this primitive returns a path Δ_3 which, when executed, links r_k and r_ℓ at $X_i \in \mathcal{X}$. This changes the configuration of robots, and places both r_k, r_ℓ in $\mathcal{R}^t_p$, $\mathit{match}^t(r_k) = r_\ell$ and $\mathit{match}^t(r_\ell) = r_k$.

4. SPLIT$(\mathbf{r})$: This primitive returns a path Δ_4 for two robots $\mathbf{r} = \{r_k, r_\ell\}$, $\mathbf{r} \subseteq \mathcal{R}^t_p$, to split their previously joined tails at their current location. The operator is the inverse of DOCK. When executed, unlinks r_k and r_ℓ at $X_i \in \mathcal{X}$. This changes the configuration of

robots, and places both r_k, r_l in $\mathcal{R}_s^t$, $\mathit{match}^t(r_k) = r_k$ and $\mathit{match}^t(r_\ell) = r_\ell$.

If the requirements on $\mathbf{r}$ are not met in 3. and 4., the SPLIT or DOCK primitive cannot be used in that context. The primitives are sequenced in triples to give high-level operators which transform world configurations. We write $\pi^t(\mathbf{r}, X_i, X_j)$, for a sequence that moves the objects at X_i to X_j: either $\pi^t(\mathbf{r}, X_i, X_j) = [\Delta_4, \Delta_1, \Delta_2]$ for $|\mathbf{r}| = 1$, or otherwise $\pi^t(\mathbf{r}, X_i, X_j) = [\Delta_3, \Delta_1, \Delta_2]$ for a pair of robots, i.e. $|\mathbf{r}| = 2$, if such collision-free paths exist and can be found by our motion planner. In the preceding, the Δ_x's give paths for the subset of robots involved, namely, $\mathbf{r}$. When $\pi^t(\cdot, \cdot)$'s have been constructed for all m robots in $\mathcal{R}$, which we write as $\Pi^t(\cdot, \cdot)$, the result is a path in the joint C-space for the team.

Let J be a deterministic cost function estimating, say, the time to execute a path. Then, to define a collective cost $\mathcal{J}$, we lift J to Π^t:

$$\mathcal{J}(\Pi^t) \;=\; \underset{\pi^t(\mathbf{r},X_i,X_j)\in\Pi^t}{\textsc{Combine}} \begin{cases} J(\Delta_4) + J(\Delta_1) + J(\Delta_2) & \text{if } |\mathbf{r}| = 1, \\ J(\Delta_3) + J(\Delta_1) + J(\Delta_2) & \text{if } |\mathbf{r}| = 2, \end{cases} \tag{4.2}$$

where COMBINE is a function that describes the way concurrent operations are aggregated for the optimization objective being considered. We use COMBINE in two ways:

- COMBINE = $\max$ for robots that stay idle until others complete their tasks.
- COMBINE = $\sum$ for computing a cumulative total navigation distance for all robots as the cost function.

A few additional elements must be introduced into the model. Firstly, let L^i be the tail length, a physical parameter, for each robot i. Secondly, we define two predicates to deal

with towing constraints as follows:

$$\begin{cases} C_1 : \mathcal{R} \times 2^{\mathcal{O}} \to \{\text{True, False}\}, \\ C_2 : \mathcal{R} \times \mathcal{R} \times 2^{\mathcal{O}} \to \{\text{True, False}\}, \end{cases} \tag{4.3}$$

where $C_1(r_i, \mathcal{O}_p = \{o_1, \ldots, o_k\})$ is true if hooking primitives by a robot r_i can move all objects in $\mathcal{O}_p$ and $C_2(r_i, r_j, \mathcal{O}_p = \{o_1, \ldots, o_k\})$ is true if dragging primitives by pairs r_i and r_j can move all objects in $\mathcal{O}_p$.

The following makes explicit assumptions which have been tacit up to this point:

(i) All our mobile robots have an inelastic tail.

(ii) Robots can link and unlink the ends of their tails, but only to form a pair.

(iii) The tail cannot pass through any of the objects when the tail contacts the objects.

(iv) The robots' and the objects' states are observable, while the tail configurations are not.

In the work hereafter, we are not interested in infeasible plans and so consider systems where there is at least one robot (or one pair of robots) that can manipulate objects, ultimately can lead to satisfying $\mathbf{F}$.

General Problem Definition: A Multi-Object Collection via Cooperative Towing (MOCCT) Planning Problem: Given $\mathcal{O}, \mathcal{R}, \mathcal{G}, \mathcal{X}, \mathcal{D}, J, \mathfrak{L}_{\text{static}}, \mathfrak{L}_{\text{pos}}$, and $\mathbf{F}$, COMBINE, with C_1 and C_2 for all robots, compute a sequence of paths $(\bar{\Pi}^0, \ldots, \bar{\Pi}^T)$ which minimizes the total delivery cost, where the capacity constraints are never violated and the goal predicate $\mathbf{F}$, is satisfied in the configuration at time T. Formally:

$$[\bar{\Pi}^0, \ldots, \bar{\Pi}^T] = \underset{[\Pi^0, \ldots, \Pi^T]}{\operatorname{argmin}} \sum_{t=0}^{T} \mathcal{J}(\Pi^t), \tag{4.4}$$

subject to

$$\models_{\mathcal{I}^T} \mathbf{F}, \quad (4.5)$$

$$T \leq \bar{T}, \quad (4.6)$$

$$\left\{ \begin{array}{ll} C_1(r_k, b^t_{X_i}) = \text{True}, & \mathbf{r} = \{r_k\} \\ C_2(r_k, r_p, b^t_{X_i}) = \text{True}, & \mathbf{r} = \{r_k, r_p\} \end{array} \right\}, \forall t \in \{0, \ldots, T\}, \forall \pi^t(\mathbf{r}, X_i, X_j) \in \Pi^t. \quad (4.7)$$

The discrete formulation leads to a single-objective combinatorial optimization problem, where the search space contains the feasible, cooperative paths for all robots.

4.3 NP-hardness of the MOCCT Problem

We show that, quite separately from the satisfiability of $\mathbf{F}$, the combinatorics of the preceding optimization problem is NP-hard. To do so we define a Trivially Satisfiable Multi-Object Collection via Cooperative Towing problem (TS-MOCCT), which is simplified in that all objects are to be moved to a single goal. Thus, the logical formula for TS-MOCCT has the special trivial structure $\mathbf{F}_{TS} := \forall i, \text{LOCATEDAT}(o_i, X_1^G)$.

To show the hardness of the TS-MOCCT problem, we will show that (i) an instance of the Vehicle Routing Problem (VRP), known to be NP-hard [109], can be reduced to an instance of TS-MOCCT, and (ii) an optimal TS-MOCCT solution can be used to generate an optimal VRP solution.

We describe the VRP [47, 109], briefly: an amount of some commodity d_i is to be delivered to each customer $o'_i \in O$ where $i \in \{1, \ldots, n'\}$ from a central depot g'_0 using m' independent delivery vehicles $R = \{r'_1, \ldots, r'_{m'}\}$. Let p_i be the position of customer o'_i. The central depot g'_0 is at p_0. Delivery is to be accomplished with the minimum total cost. $\mathcal{J}'(p_i, p_j)$ denotes the transit cost from p_i to p_j. The goal of VRP is to find a partition of n' customers into m' cycles $\{s_1, \ldots, s_{m'}\}$ whose only intersection is the depot node

(starting and ending at the central depot). Overall, VRP = $\langle \mathcal{O}' = \{o'_1, \ldots, o'_{n'}\}, \mathcal{R}' = \{v_1, \ldots, v_{m'}\}, \mathcal{G}' = \{g'_0\}, \mathcal{X}' = \{p_0, p_1, \ldots, p_{n'}\}, \mathcal{D}' = \{d_1, \ldots, d_{n'}\}, \mathcal{J}' \rangle$

Lemma 1 *The* TS-MOCCT *problem is NP-hard.*

Proof : We give a polynomial-time transformation of VRP $= \langle \mathcal{O}', \mathcal{R}', \mathcal{G}', \mathcal{X}', \mathcal{D}', \mathcal{J}' \rangle$ into the TS-MOCCT problem $= \langle \mathcal{O}, \mathcal{R}, \mathcal{G}, \mathcal{X}, \mathcal{D}, C_1, C_2, J, Combine \rangle$ We map VRP inputs to an instance of the MOCCT problem as follows:

1. $n = n'$, then $\mathcal{O} = \mathcal{O}'$ and $\mathcal{D} = \mathcal{D}'$.
2. We set $C_2 =$ true and $C_1 =$ false for all robots.
3. We consider only pairs of robots, which was already enforced by picking $C_1 =$ false. We let $m = 2m'$, so $\mathcal{R} = \{r_1, \cdots, r_{2m'}\}$.
4. We have a single goal, then $\mathcal{G} = \{g'_0\}$ for the TS-MOCCT problem.
5. We have the set of positions $\mathcal{X} = \{X_1, \ldots, X_{n'}, X_1^R, \ldots, X_m^R, X_1^G\}$, where $X_i = p_i$ for $i \in \{1, \ldots, n'\}$. The robots' initial locations X_i^R map to p_0 for all i. The goal location X_1^G maps to p_0.
6. The cost function from X_i to X_j for a pair of robots is $J(\Delta_3) + J(\Delta_1) + J(\Delta_2)$, which results in the collective cost, $\mathcal{J}(X_i, X_j)$, mapping to $\mathcal{J}'(p_i, p_j)$.
7. We set COMBINE = $\sum$ for computing a total navigation distance for all robots.

This transformation defines all inputs of the VRP problem in terms of the TS-MOCCT problem.

Next, we show that an optimal TS-MOCCT solution corresponds to the optimal VRP solution. We consider the optimal TS-MOCCT solution $\{\bar{\Pi}^0, \ldots, \bar{\Pi}^T\}$ for m' pairs of robots. The optimal solution gives that each pair of robots can start and end at the goal location.

No optimal solution involves any pair of robots returning to the goal during traveling (except the start and the end). Recall that the TS-MOCCT problem imposes the constraint (as a modeling assumption) that all the objects at X must be moved together. Thus, the TS-MOCCT solution consists of m' cycles (potentially empty) for each pair of robots with no repeated nodes except the goal location X_1^G. We rewrite the solution in terms of the m' cycles:

$$\begin{aligned}[\bar{\Pi}^0, \ldots, \bar{\Pi}^T] = [\bar{\Pi}_1(X_0, X_i)^0, \ldots, \bar{\Pi}_1(X_j, X_0)^T], \ldots, \\ [\bar{\Pi}_{m'}(X_0, X_k)^0, \ldots, \bar{\Pi}_{m'}(X_p, X_0)^T]\end{aligned} \tag{4.8}$$

Since we have m' number of pairs, we can see that the cost functions are equivalent when we rewrite the summation over cycles.

$$\sum_{k=1}^{m'} \sum_{X_i, X_j \in \bar{\Pi}_k} \mathcal{J}(\bar{\Pi}_k(X_i, X_j)) = \sum_{k=1}^{m'} \sum_{(p_i, p_j) \in s_k} \mathcal{J}'_k(p_i, p_j). \tag{4.9}$$

Thus, an optimal solution of the TS-MOCCT problem must also be an optimal solution of the VRP.

Theorem 1 *The* MOCCT *problem is NP-hard*

Proof : From Lemma 1, we proved the TS-MOCCT problem is NP-hard. The MOCCT problem is the generalized version of the TS-MOCCT, which can have more goal locations with complex goal conditions,TS-MOCCT $\subseteq$ MOCCT. Thus the MOCCT problem is NP-hard.

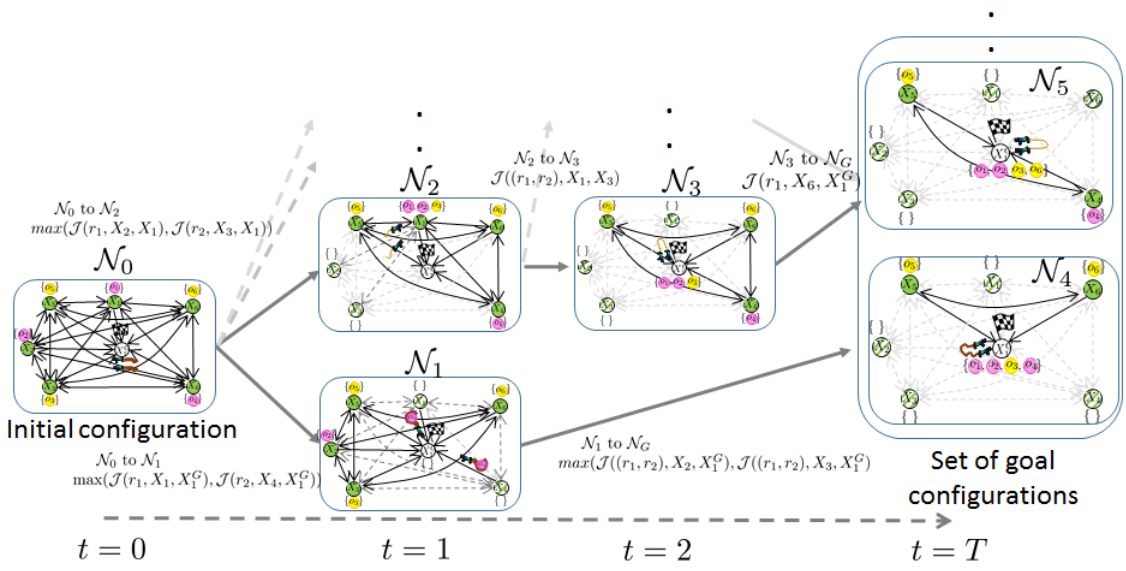


Figure 4.3: Construction of a plan by searching a tree from the left ($t = 0$) to the right ($t = T$). Each node consists of $\mathcal{O}$, $\mathcal{R}$, and $\mathcal{G}$.

4.4 Algorithms for the MOCCT problem

Searching the space of all possible motion combinations for multiple robots and objects is prohibitive. Consequently, we are compelled to introduce heuristic simplifications. In this section, we describe two algorithms we used for the MOCCT problem:

(1) We seek to produce solutions of reasonable quality in limited time via A* search. The planner minimizes time to complete the task by searching over the full solution space, but is guided by heuristics.

(2) We propose a more efficient algorithm which prunes some of the choices available to the algorithm in (1), but which risks potentially overlooking plans with minimum cost.

4.4.1 The Basic Heuristic Search Algorithm

The algorithms operate on a tree structure describing the objects and robots configuration: each node has the robots' current states (positions in $\mathcal{X}$ and *match*$^t(\cdot)$), and the objects' current states (position in $\mathcal{X}$). Figure 4.3 shows how the search tree is constructed by using the example of Figure 3.1. The initial configuration $\mathcal{N}_0$ (shown in Figure 4.2(a)) is the root node in Figure 4.3. Then, we expand a tree node based on all possible choices satisfying the constraints (*e.g.,* tail length and capacities).

An example helps to understand the expansion of tree nodes: take the two sequential steps of Figure 4.1. Step-1 shows that r_1 transfers o_1 at X_1 to the goal, X_1^G, while r_2 transfers o_4 at X_4 to the goal, X_1^G. This takes time transition from $\mathcal{N}_0$ to $\mathcal{N}_1$ in Figure 4.3, where the transition cost is $\max(\mathcal{J}(r_1, X_1, X_1^G), \mathcal{J}(r_2, X_4, X_1^G))$. Step-2 shows a pair of robots (r_1, r_2) transferring b_{X_2} and b_{X_3} to the goal, that is, a transition from $\mathcal{N}_1$ to $\mathcal{N}_4$ in Figure 4.3, with cost $\max(\mathcal{J}\left((r_1, r_2), X_2, X_1^G\right), \mathcal{J}\left((r_1, r_2), X_3, X_1^G\right))$.

We see that there exist a set of goal configurations satisfying $\mathbf{F}$ (in this problem instance we require four objects at the goal location, with at least two objects that are pink). For example, $\mathcal{N}_0$, $\mathcal{N}_2$, $\mathcal{N}_3$, and $\mathcal{N}_4$ are transitions that reach another goal configuration in Figure 4.3.

Assuming the towing capacity (maximum mass) is not exceeded, a pair of robots operating together is capable of moving more than one set of objects at multiple locations to X_j. In executing Step-2 from Figure 4.2(d), the pair move o_2 from X_2 and o_3 from X_3 to X_1^G in one action. The search tree must include these kinds of actions because they represent part of the real cost savings that one obtains from pairs. Thus, to add extra such edges departing X_i, we look for nearby X_j's, sorting these in ascending order by Euclidean distance. Let $\hat{B}_{X_i}$ be the set of objects nearby X_i. Here "nearby" includes only those no more than the length of the tail away.

Figure 4.4 gives an example of how these nearby objects introduce new choices. Assume the pair pick a route to transfer object(s) at X_1 to the goal X_1^G. There are four subsets of $\hat{\mathcal{B}}_{X_1}$ (red, blue, green, and purple dotted lines). For example, the red dotted line shows one sibling, indicating $\hat{\mathcal{B}}_{X_1} = [o_1, o_5]$, in ascending order. Similarly, the blue dotted line shows three siblings, indicating $\hat{\mathcal{B}}_{X_1} = [o_1, o_5, o_4]$. The purple dotted line shows five siblings, $\hat{\mathcal{B}}_{X_1} = [o_1, o_5, o_4, o_3, o_2]$. We define a cost for this action, which we take to be the perimeter of the convex hull of $\hat{B}_{X_i}$. In Figure 4.4, the cost of the purple subset is the total distance of the outer positions X_1, X_2, and X_3. The transfer cost for o_4 and o_5 is zero because they are inside of the convex hull, so are transported effectively for free. Similarly, the cost of the green subset is the total distance among X_1, X_4, and X_3, and no cost is incurred for o_5.

The basis of Algorithm 5 is A* search, extended to include an implicit search for these additional nearby objects. The first phase, Line 4, computes the set of nearby objects, $\hat{\mathcal{B}}_{X_i}$, for all $X_i \in \mathcal{X}$, satisfying the tail length constraints and the capacity constraints of robots. Then, we start to enumerate the possible sets (Lines 12 to 20). Given m robots, we construct the possible combinations of either role (independent or as half a pair). Then, we find all combinations of choices for $\mathbf{r}$, called $\hat{A} \in A$, which a kind of satisfy the capacity

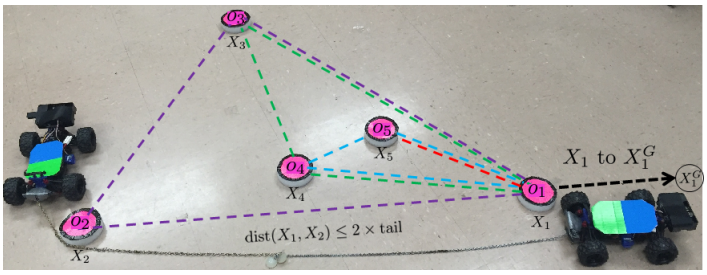


Figure 4.4: Five objects induce several choices. There are several subsets of $\hat{\mathcal{B}}_{X_1}$: The blue dot line shows $\{b_{X_5}, b_{X_4}\}$. The green dot line shows $\{b_{X_5}, b_{X_4}, b_{X_3}\}$. The purple dot line shows $\{b_{X_5}, b_{X_4}, b_{X_3}, b_{X_2}\}$.

Algorithm 5 Basic_Search

INPUT: $\mathcal{O}, \mathcal{R}, \mathcal{G}, \mathcal{X}, \mathcal{J}, \mathcal{R}, \mathcal{D}, \mathbf{F}$
OUTPUT: $(\bar{\Pi}^0, \ldots, \bar{\Pi}^K)$
Q = $\mathcal{N}_0 = [pos^0(\mathcal{O}) \cup pos^0(\mathcal{R}) \cup pos^0(\mathcal{G})]$ and ClosedSet = $\emptyset$
Compute $\hat{\mathcal{B}}_{X_i}, \forall i$
while Q $\neq \emptyset$ **do**
  $\mathcal{N}_{curr} = POP(Q)$
  if $\mathbf{F} =$ true **then**
    return $(\bar{\Pi}^0, \ldots, \bar{\Pi}^K)$ from $\mathcal{N}_0$ to $\mathcal{N}_{curr}$
    break
  end if
  ClosedSet.add($curr$)
  for each of possible robot settings **r** of $\mathcal{R}$ **do**
    for each possible motions $\hat{A}$ of A **do**
      for each possible B_{X_i} of $\hat{\mathcal{B}}_{X_i}$ **do**
        $\mathcal{J}$ = Assignment(**r**,$\hat{A} \cup B_{X_i}$)
        $\mathcal{N}_{near}$.cost =max($\mathcal{J}$) + h($\mathcal{N}_{near}$, X_k^G)
        Q = Q $\cup \mathcal{N}_{near}$.
      end for
    end for
  end for
end while

constraints in Equation (4.7). In Line 14, we increase the set by including nearby objects (following the sequential ordering of $\hat{\mathcal{B}}_{X_i}$) until the maximum capacity C_2 is exceeded, and $\mathbf{F}$ is impossible to achieve. For example, in Figure 4.4, we pick the basis set $\{o_1\}$ first, and then increase the range by accumulating an element of $\hat{\mathcal{B}}_{X_i}$ to have the red area $[o_1, o_5]$, the blue area $[o_1, o_5, o_4]$, then the green one $[o_1, o_5, o_4, o_3]$, and finally the purple set $[o_1, o_5, o_4, o_3, o_2]$. In Line 15, we have to match a set of robots with a set of choices. We use the Hungarian method [110] to find the optimal assignment between them. In Line 16, the maximum (not sum of) distances of $\mathbf{r}$ at the current step is used because all the robots are working in parallel, and we add the heuristic estimate: the total distance between X_i and goals, $\sum_{i\in[1,n]} \min_{j\in[1,k]}(\boldsymbol{pos}^t(o_i), X_j^G)$.

4.4.2 The Opportunistic Neighborhood Search (ONS) Algorithm

To reduce the search cost, we reduce the number of outgoing choices induced by the set operation on nearby objects just described. We do this by being greedy in a manner we feel represents a kind of opportunism. This algorithm is a modification of the basic algorithm, in which we do not search over all possible subsets of objects in Line 14, instead we pick the maximum sized set of objects satisfying constraints. This is efficient if we have objects that are located within the boundary of the tail length.

4.5 Experiments

The basic heuristic search (called *Basic*) and the opportunistic neighborhood search (called *ONS*) algorithms produce solutions for moderate numbers of objects and robots in a reasonable time. The *ONS* algorithm produces solutions for even larger number of objects and robots, though precise difference in costs depend on the constraints (*e.g.,* a longer tail with larger capacities). All algorithms were executed on an Intel Core i7 3.2GHz, and implemented in Matlab with a SMT-solver [90]. For comparison purposes, we also implemented an uninformed breadth first search, which we call *Exact*.

We have three goal predicates to validate our approach:

1. $\mathbf{F}_1$: transferring all PINK and CYLINDER objects to X_1^G.

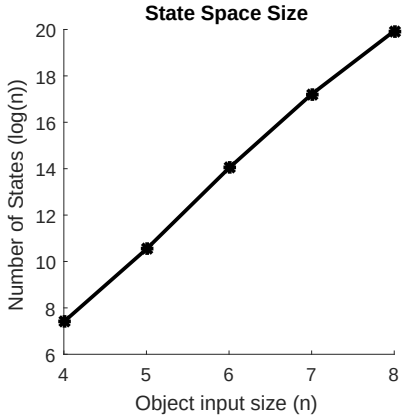
$$\mathbf{F}_1 := \forall i, \textsc{pink}(o_i) \wedge \textsc{cylinder}(o_i) \wedge \textsc{locatedAt}(o_i, X_1^G) \tag{4.10}$$

2. $\mathbf{F}_2$: transferring four objects to X_1^G, where two objects are at least pink and $\textsc{num}(X)$ returns the number of objects at X.

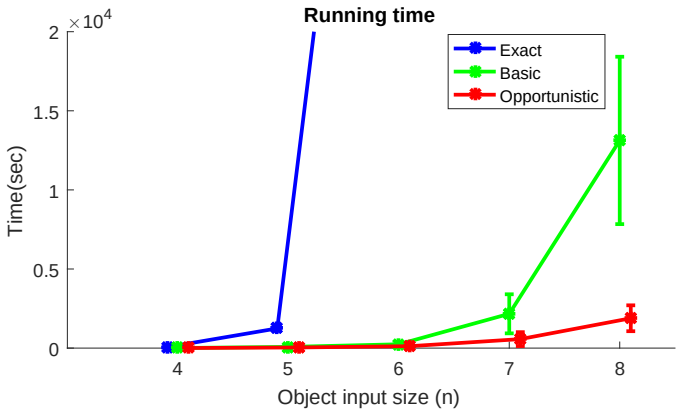
$$\begin{aligned}\mathbf{F}_2 := \forall i, \forall j, i \neq j, \textsc{pink}(o_i) \wedge \textsc{pink}(o_j) \wedge \textsc{locatedAt}(o_i, X_1^G) \\ \wedge \textsc{locatedAt}(o_j, X_1^G) \wedge \textsc{num}(X_1^G) = 4\end{aligned} \tag{4.11}$$

3. $\mathbf{F}_3$: transferring objects to X_1^G, X_2^G, or X_3^G, such that $0 < \textsc{num}(X_2^G) < \textsc{num}(X_1^G)$ and $\textsc{num}(X_3^G) = \textsc{num}(X_1^G) + \textsc{num}(X_2^G)$.

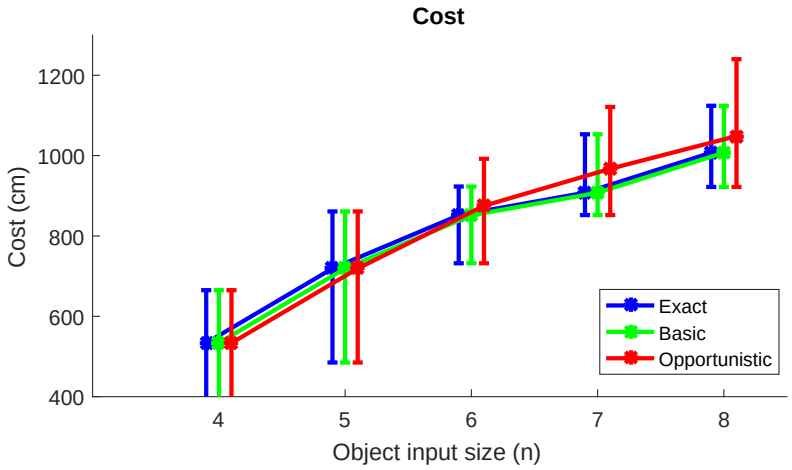
$$\begin{aligned}P_1 &:= 0 < \textsc{num}(X_2^G) < \textsc{num}(X_1^G) \\ P_2 &:= \textsc{num}(X_3^G) = \textsc{num}(X_2^G) + \textsc{num}(X_1^G) \\ \mathbf{F}_3 &:= P_1 \wedge P_2\end{aligned} \tag{4.12}$$



(a) State-space sizes for five settings.



(b) Running times for five settings.



(c) Solution Costs for five settings.

Figure 4.5: State-space sizes, running times, and solution costs of the MOCCT problem.

The formulation given on page 65 involved modeling robot towing feasibility with two predicates, C_1 for single robots, C_2 for pairs. In our physical implementation, we used simple instances of these two aspects. We assume identical tails, so treat all physical parameters (*e.g.,* length, mass, the coefficient of friction, etc.) as the same. We use a chain for the tail, so impose no curvature constraint— *i.e.,* the smallest radius is zero. Also, in our experiments, all objects have identical masses and shapes. Using all the preceding simplifications if suffices to have a condition that depends only on the number of objects.

Thus we define $C_1(r_i, \mathcal{O}_p) = (|\mathcal{O}_p| \leq \bar{C_1})$ and $C_2(r_i, r_j, \mathcal{O}_p) = (|\mathcal{O}_p| \leq \bar{C_2})$. In other words, $\bar{C_1}$ is the maximum capacity for a single robot and $\bar{C_2}$ is the maximum capacity for a pair of robots.

We use $\mathbf{F}_1$ from Section 4.5.1 to Section 4.5.3. First, we show how the combinatorial aspects of the MOCCT problem means it has an enormous state-space even with few objects. See the plot in Figure 4.5(a). For example, even 6 objects with fixed parameters (a tail length $L = 100\ cm$, $m = 2$, $\bar{C_1} = 1$, $\bar{C_2} = 3$) has over one million states.

Second, we evaluate the efficiency and performance of our algorithms by randomly generating environments and measuring plan cost and computational time. The evaluation was conducted with the following fixed parameters: two tail robots, $\bar{C_1} = 1$, $\bar{C_2} = 3$, and $L = 100\ cm$.

And then, we use $\mathbf{F}_2$ and $\mathbf{F}_3$ to demonstrate how our planner solves the complex tasks in Section 4.5.4.

4.5.1 Random Environments

Our simulated environments are $3\ m \times 3\ m$ open spaced regions. We have several parameters: the number of objects n, the number of robots m, maximum capacities for C_1 and C_2 functions, and the tail length L. To test the algorithms, we increased n from 4 to 8 with fixed number of robots $m = 2$. We generated 10 random environments for each number of objects.

4.5.2 Evaluation of Algorithms

As visible in Figure 4.5(b), The *Basic* finds an optimal solution in a reasonable time on small instances ($n \leq 6$); even a small problem has a huge search space (approximately a hundred thousand states when $n = 8$). In our tests, The *ONS* reduces a search space as reflected in its running-time improved, while the quality of solutions it finds is acceptable (see Figure 4.5(c)).

4.5.3 Physical Robot Experiments

4.5.3.1 System Setup

We demonstrate that the proposed solution works in practice. We used two RC cars, controllable at velocities approximately $0.4\ m/s$. An embedded computer on the car controls the robot and communicates with a separate computer integrated with an overhead tracking system which localizes the objects and the robots. Our software uses the ROS framework. All experiments are conducted in our test arena of size $5\ m \times 5\ m$.

4.5.3.2 Robot Motion Model

We use a simple car model for the robot same as Section 3.

For a single robot, we use a hooking primitive in the TRANSIT$(\cdot)$ primitive, which sets up the robot's orientation to work on the TRANSFER$(\cdot)$ primitive. To produce this hooking primitive, we put a fist-size quarter sphere at the end of the tail. First, the robot winds around the object, crossing near the end of its tail (Figure 3.18(b)). Then, the quarter sphere works like a buckle (Figure 3.18(c)). Thereafter, the robot moves the hooked object via the TRANSFER$(\cdot)$ primitive. To release hooked objects, the robot turns around to the object reversing the direction of the winding motions (Figure 3.18(d) to Figure 3.18(f)).

For a pair of robots, we developed a DOCK$(\cdot)$ primitive. Firstly, given a subtask (move objects from X_i to X_j), one robot stays near X_i, initializing the robot's orientation for the TRANSFER$(\cdot)$ primitive (Figure 3.17(a)), then the second robot crosses the tail of the first (Figure 3.17(b)). Finally, the tails become connected (because each tail contains a magnet) as the robot follows the contour of the convex hull consisting of the outer positions X_i. See Figures 3.17(b) and 3.17(c). After that, we use TRANSFER$(\cdot)$ primitive to move the gathered objects. To split conjoined tails, we simply make an extra motion: one robot stops while the other moves past until the length of the tails is exceeded, breaking their connection.

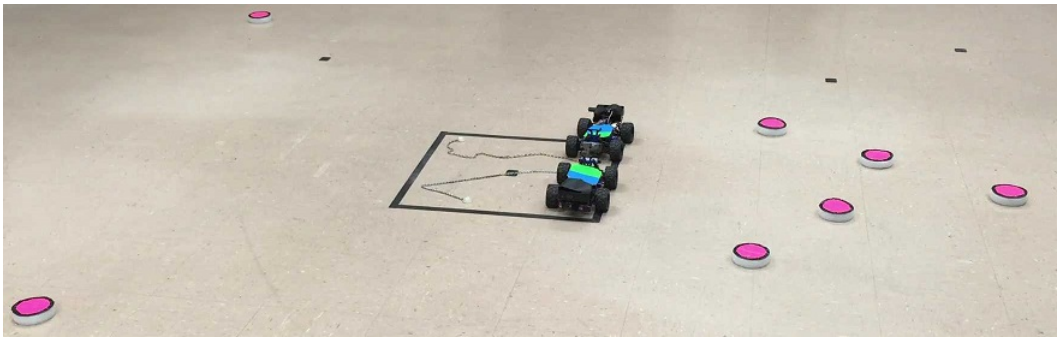


Figure 4.6: We have seven pink cylindrical objects to be towed to the center of the room with two robots.

4.5.3.3 Three Planners

For experimental comparison, we considered three kinds of planners:

(1) The *single-only* planner uses only two single robots.

(2) The *pair-only* planner uses only one pair of robots.

(3) The *both* planner uses both roles of robots.

4.5.3.4 Experimental Validation Planners

To validate our model, we set $n = 7$ objects with two tail robots where $\bar{C}_1 = 1$, $\bar{C}_2 = 5$, and $L = 100\ cm$. Figure 4.6 shows the basic settings for all planners. All types of planners use the ONS algorithm to compute their solution: The planning execution time of (i) takes 349 seconds with 4404 states, while (ii) takes 123 seconds with 1684 states and (iii) takes 24 seconds with 691 states.

Figures 4.8 to 4.10 show the single result of the three different planners: *single-only*, *pair-only*, and *both* as shown in the first row, the second row, and the third row, respectively. For each row, from left to right, we see the snapshots of the physical experiments over time.

The *single-only* planner uses two robots acting individually and simultaneously. There are $K = 4$ transitions in the solution. In addition to that, the *single-only* planner completes this task in 285 seconds. The snapshot at the arrival time is shown in Figure 4.11(a). The blue lines indicate the robots' trajectories and the pink lines are the objects' trajectories, showing how the robots work and the objects are moved. Since each robot needs space to release a hooked object, the final position of objects is distributed near the goal boundary.

The *pair-only* planner uses that a pair of robots are bringing each object together. Firstly, they move a single object at upper-left shown in Figure 4.9(a). Next, they move a single object at lower-left shown in Figure 4.9(b). Finally, they move five objects together in Figure 4.9(d). Figure 4.9(c) shows the releasing of the gathered objects. There are the $K = 3$ transitions in the solution. This solution of the *pair-only* planner completes the task in 225 seconds. The snapshot at the arrival time appears in Figure 4.11(b).

The *both* planner shows that two robots first manipulate objects individually by using a hook primitive, and then the robots connect their tails and cooperatively drag multiple objects. There are the $K = 2$ transitions in the solution. Firstly, two robots move objects at upper-left and lower-left shown in Figure 4.10(a), and then they release objects at the goal (Figure 4.10(b)) individually (but simultaneously). Lastly, they use a docking primitive (Figure 4.10(c)), and drag five objects to the goal together (Figure 4.10(d)). This solution of the *both* planner completes the task in 223 seconds. The snapshot at the final time is in Figure 4.11(c).

A detailed analysis of all the results appears in Figure 4.7, shows how the algorithm's expected costs are good predictions of reality. The blue bar is the estimate cost (cm) obtained by the proposed algorithm, while the red bar is a real cost (sec) based on real test with ten trials. Obviously calibration could relate these two cost metrics—but the data show that ordering between plans is already satisfaction. We observe that the *pair robots-only* planner and the *both* planner do not have a large disparity from reality. The reason

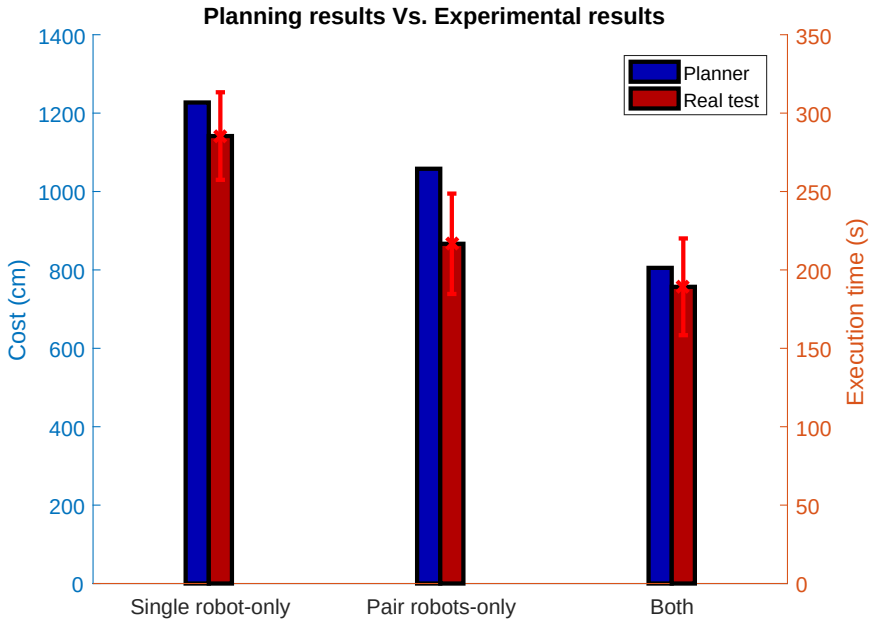
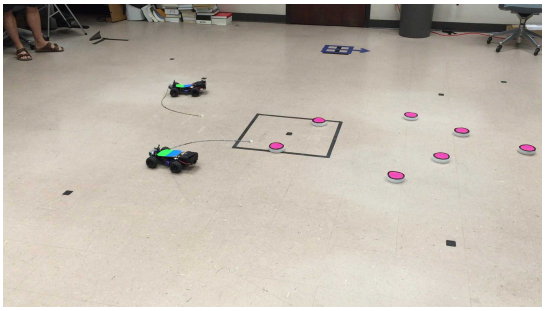


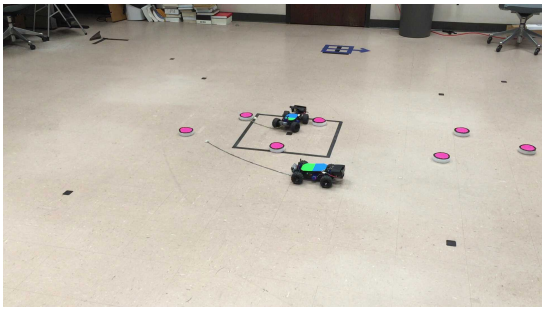
Figure 4.7: Planner and physical robot experimental results.

is that the *both* planner must pay the cost of the docking procedure, but *pair-only* planner need not execute the DOCK$(\cdot)$ primitive.

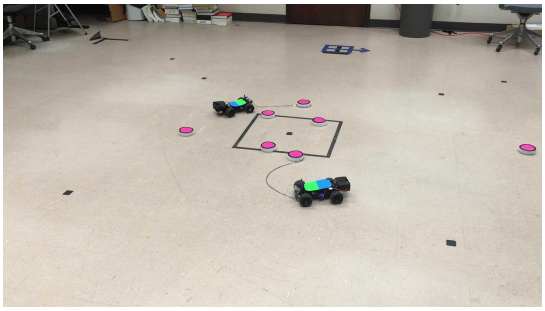
In addition to that, the primitives that the pairs of robots use are simpler and more efficient to realize in practice than the hooking and unhooking operations.



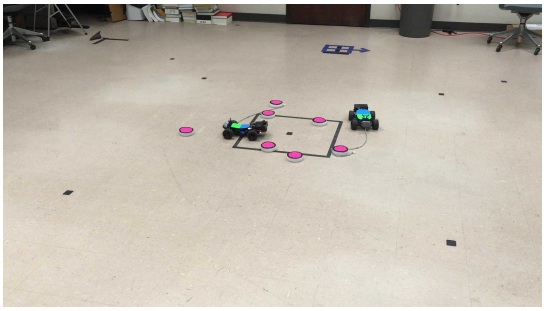
(a) The *single-only* at 97 seconds: two single tail robots release the hooked objects individually.



(b) The *single-only* at 134 seconds: two single tail robots release the next objects individually.

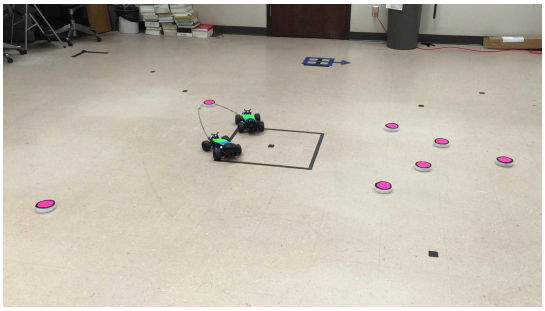


(c) The *single-only* at 238 seconds: two single tail robots release the next object.

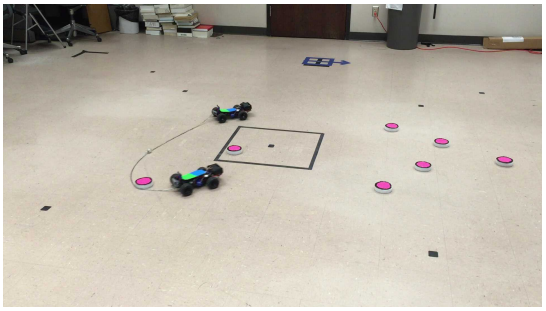


(d) The *single-only* at 280 seconds: one single tail robot releases the hooked object alone at the goal.

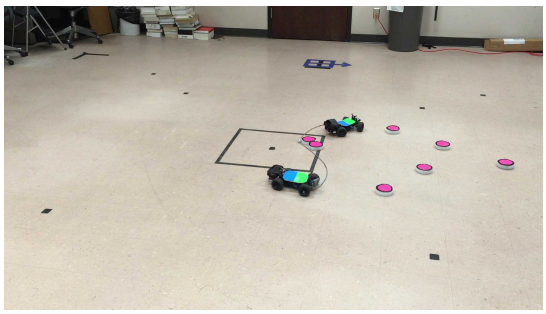
Figure 4.8: The *single-only* planner.



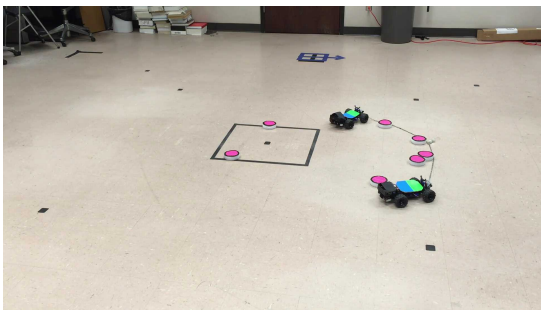
(a) The *pair-only* at 70 seconds: one pair of robots moves one object (upper-left) to the goal together.



(b) The *pair-only* at 110 seconds: one pair of robots moves one object (lower-left) to the goal together.

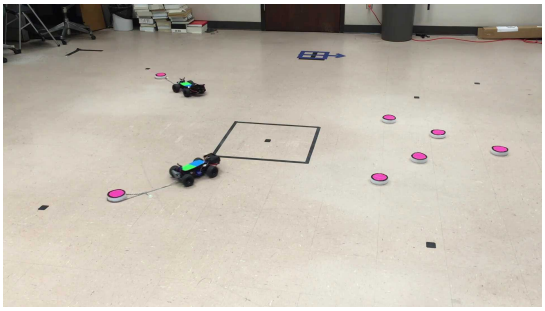


(c) The *pair-only* at 143 seconds: this shows the procedure of releasing objects by a pair of robots.

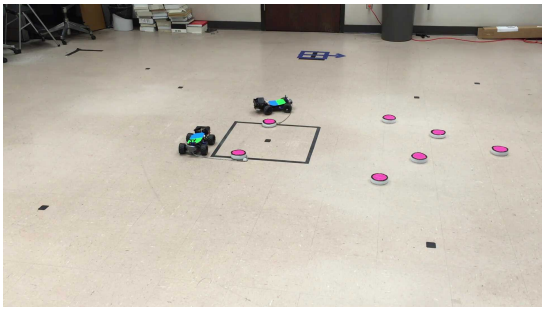


(d) The *pair-only* at 216 seconds: one pair of robots move five objects to the goal together.

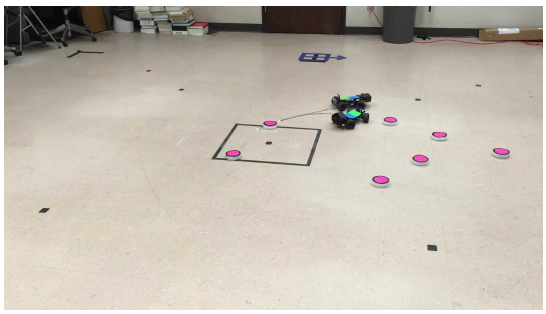
Figure 4.9: The *pair-only* planner.



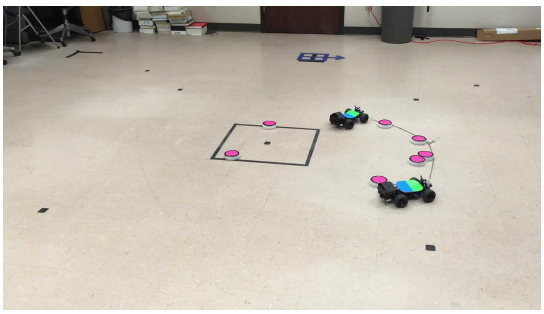
(a) The *both* at 70 seconds: two single tail robots move two objects individually.



(b) The *both* at 106 seconds: two single tail robots release two objects individually using an unhooking primitive.

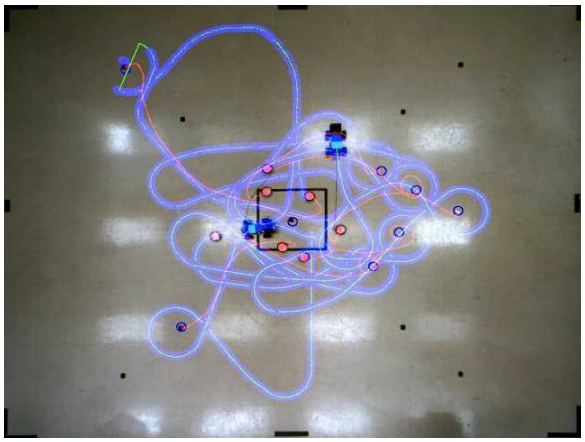


(c) The *both* at 160 seconds: two single tail robots use a docking primitive, and then conjoined via the contact of each tail.

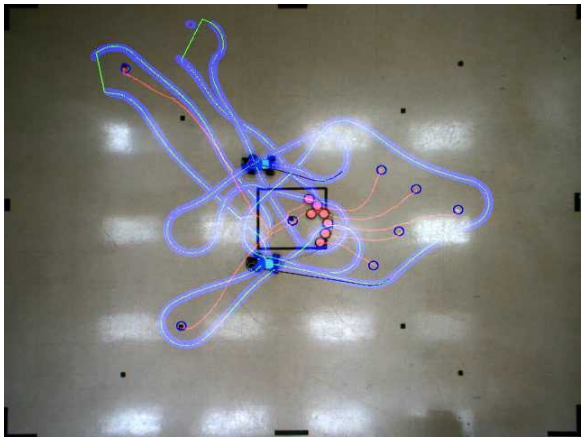


(d) The *both* at 210 seconds: a conjoined two robots drags five objects together at the same time.

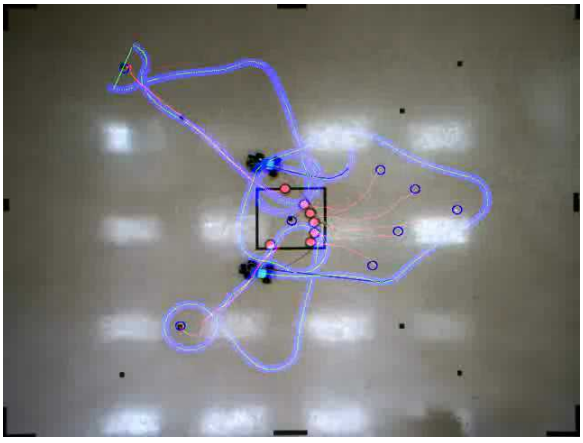
Figure 4.10: The *both* planner



(a) *single-only* completes the task at 285 seconds.



(b) *pair-only* completes the task at 225 seconds.



(c) *both* completes the task at 223 seconds.

Figure 4.11: This view is from the overhead tracking system. The accumulated trajectories for objects and robots are displayed: the blue lines are the robots, while the pink lines are the objects.

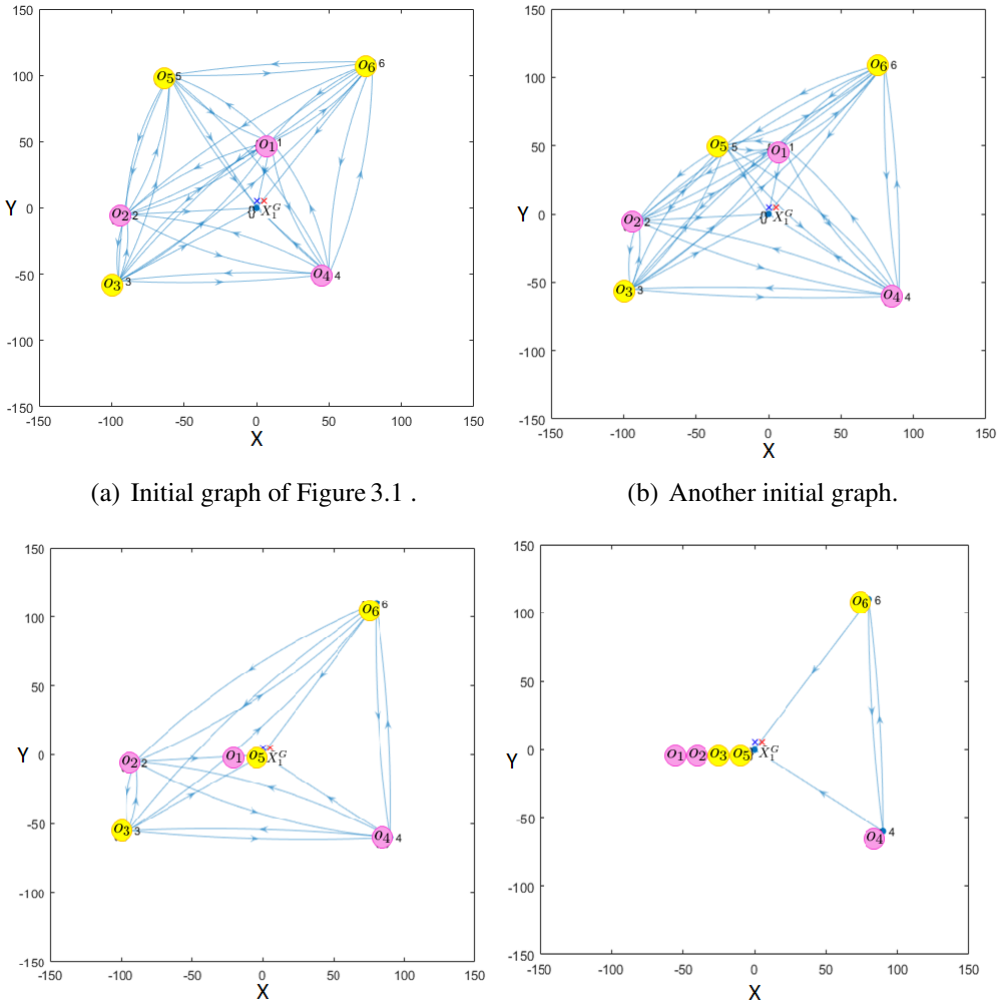
4.5.4 Other Experimental Results for Other Scenarios

We verify our planner in richer goal states using by $\mathbf{F}_2$. Here we have two robots and 6 objects (3 yellow, 3 pink) where $\bar{C}_1 = 1$, $\bar{C}_2 = 5$, and $L = 70\ cm$ for all robots. We used the ONS algorithm to demonstrate here. We showed two cases of 'scenario-2' that can have different solutions: (1) One is shown in Figure 4.12(a) as a topological graph. (2) Another is shown in Figure 4.12(b), showing that the yellow o_5 is closely located to the pink o_1, and the pink o_4 is located little far away from the goal compared to (1).

Our planning time for $\mathbf{F}_2$ took 26.4 seconds with 517 states for the first case. The initial graph of the first case is shown in Figure 4.12(a). There are three pink objects and three yellow objects with two robots and one goal X_1^G. The planning results for the first case is shown in Figures 4.2(c) and 4.2(d). First, two single robots transfer o_1 and o_4 independently, and then a pair of robots transfer o_2 and o_3 together. The initial graph of the second case is shown in Figure, 4.12(b), which took 12.5 seconds with 223 states. The results are shown in Figure 4.12. A pair of robots transfers o_1 and o_5 first (Figure 4.12(c)) to the goal, and then transfers o_2 and o_3 to the goal (Figure 4.12(d)). Both results have two transitions ($T = 2$) in the solution. Here, we can see that the change of the topological relationship among objects makes for different solutions. We also expect that several parameters such as the tail length and the maximum capacities for a pair of robots alter the solutions found. But, the detailed results are not included here.

We also used our planner on the complex goal predicate $\mathbf{F}_3$ as 'scenario-3'. It is a fairly large problem with 4 robots and 10 objects, all pink, where $\bar{C}_1 = 1$, $\bar{C}_2 = 5$, and $L = 50\ cm$ for all robots. Only the ONS algorithm was used. Our planning time for $\mathbf{F}_3$ took 1353 seconds with 4469 states. The initial graph is shown in Figure 4.13(a). The planning result is shown in Figures 4.13(b) and 4.13(c). First, two single robots transfer o_2 and o_5 to X_1^G independently, while one pair of robots transfer o_3 and o_9 to X_3^G together.

Second, one pair of robots transfers o_6 to X_2^G, while another pair of robots transfers o_7 to X_3^G. Then the final result satisfies $\mathbf{F}_3$. Here the result shows two transitions ($T = 2$) in the solution.



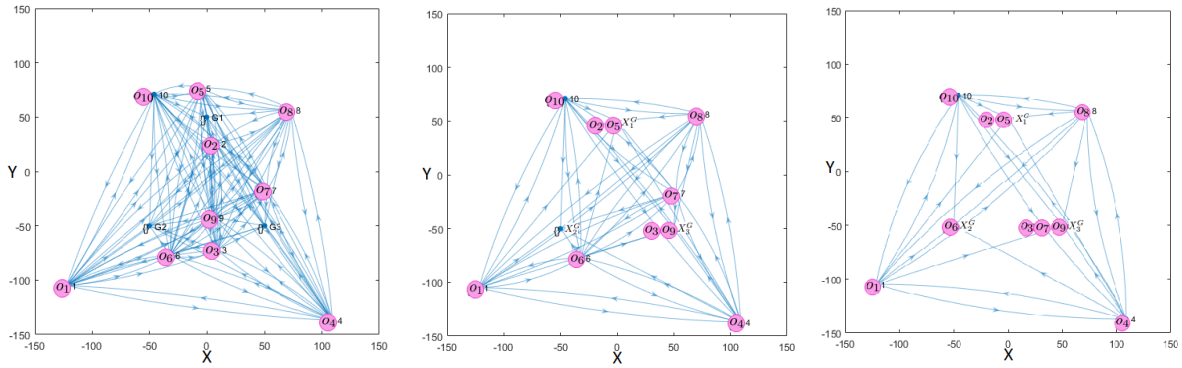
(a) Initial graph of Figure 3.1 .

(b) Another initial graph.

(c) Step-1: from (b) to (c), one pair of robots transfers o_1 and o_5 to the goal X_1^G.

(d) Step-2: from (c) to (d), one pair of robots transfers o_2 and o_3 to the goal X_1^G. This satisfies $\mathbf{F}_2$.

Figure 4.12: The simulation results are plotted using graph representation. There are three pink and three yellow objects, two robots, and one goal; (a) is for the example in Figure 4.1(a). Another example is shown in (b), which modified the topological relationship of (a). We show the results of the planner for the example (b) via (c) and (d). Finally, (d) satisfies $\mathbf{F}_2$.



(a) There are three goals,X_1^G, X_2^G, X_3^G We use four robots and ten objects. Initially, all robots are located at X_1^G.

(b) Step-1: from (a) to (b), two single robots transfers o_2 and o_5 to X_1^G individually while one pair of robots transfers o_3 and o_9 to X_3^G together.

(c) Step-2: from (b) to (c), one pair of robots transfers o_6 to X_2^G while another pair of robots transfers o_7 to X_3^G. This satisfies $\mathbf{F}_3$.

Figure 4.13: The simulation results of a graph representation; (a) shows ten pink objects and three goals. We used 4 robots located at X_1^G initially. We show the results of the planner for the example (a) via (b) and (c). Finally (c) satisfies $\mathbf{F}_3$.

4.6 Discussion

4.6.1 Obstacle-free Paths in the Topological Graph

We explored several scenarios to validate our approaches. We mentioned that we have obstacle-free path for the robots. Before the algorithm execution, we have a pre-computation step to construct the initial set of topological relationships. When we examine other possible choices, we include the choice when it satisfies a geometric constraint. This geometric constraint comes from the primitives: when the robots execute the primitives, the robots need a specific working boundary for surrounding objects, docking, and splitting two robots. If the robot's working boundary intersects objects and other static obstacles in the configuration space, then we remove the choice from X_i to X_j by $\mathbf{r}$. For example, assuming an object is located near a static obstacle. If the objects are very close to the obstacle, then the robots cannot move objects because there is insufficient space between

objects and obstacles to execute a hooking primitive. Consequently, choices must represent collision-free paths, not only for robots, but also for the length of tail that can actually manipulate objects. This helps to simplify our approach.

4.6.2 Loosely-coupled and Tightly-coupled Coordination

Discussions of multi-robot systems often uses qualifiers like 'loosely-coupled' or 'tightly-coupled' to indicate the frequency of information exchange. In the present work, when the robots operate as conjoined pairs, they are literally physically coupled so that the motions of one robot directly constrains the other. The cases where robots operate as separate individuals certainly involves less direct interaction. In this regard, this work represents an instance where the planner is automatically making decisions about whether loose or tight coordination is best, rather than having a single form of coordination predetermined beforehand.

4.6.3 How to Build a Tail for MOCCT *Problems: Guidelines for a Practitioner*

In this section, we want to discuss design criteria for the tail robot. The physical parameters for tails are not explained in detail in the preceding text in the formulation of Section 4.2.1. There we simply abstracted these parameters into the capacity constraints.

Here, we revisit terminologies first described in Section 3. There are three forces at play: the force induced by the robot F_r, the tail's static friction F_t, and the object's static friction F_o. We consider two coefficients of friction; let μ_t be the coefficient of friction between tails and the workspace floor and let μ_o be the coefficient of friction between objects and the workspace floor. Let L^i represent the tail length for a robot i and m_t be the mass of the tail per unit length. Then, the following condition must be satisfied if the robot is to move the object:

$$F_r \geq F_t + F_o, \tag{4.13}$$

where $F_t = \mu_t \cdot m_t \cdot L^i \cdot g$ and $F_o = \mu_o \cdot M_o \cdot g$.

The predicate C_1 is also determined by the perimeter of the object boundaries d_o (*cf.* Figure 3.18 on page 46). The hooking and releasing primitives need the extra length d_r to surround and release the secured object owing to the extent of the robot (*cf.* Figures 3.18 and 3.19 on pages 46 and 47). Then, to drag objects with a single robot, the following second condition for the tail length L^i must be satisfied:

$$L^i \geq d_o + d_r. \tag{4.14}$$

The predicate C_2 depends on: (1) the force induced by two robots, (2) the perimeter of the clustered object boundaries, (3) the total length of two conjoined tails. Then, Equation (4.13) can be modified as follows: F_r is for two robots and F_t is for two tails. Equation (4.14) can also be modified as follows: Have d_o be the perimeter of the convex hull of the clustered objects, and d_r can be extended for two robots, analogously.

Consequentially, Equation (4.14) gives the lower bound by L^i because there are relationships among length of tails, mass of tails, and forces produced robots. Equation (4.13) gives the upper bound depending on the perimeter of the object boundaries and the length of tails.

However, in our experience, a tail length should be more moderate than the maximum length bound in Equation (4.13). For example, if robots have tails that are excessively long, then the initializing steps for hooking, releasing, and docking primitives can be infeasible when obstacles hinder motions. In such cases the possible actions from X_i to X_j will be removed by the planner, and the problem could be found to be infeasible.

Similarly, tail stiffness can be considered as a curvature constraint. Let κ be the nonzero radius of the smallest circle which can be bent (see Teshnizi and Shell [111]). This requires tails to have at least $2\pi\kappa$: (1) one robot should wrap objects by hooking. (2) a pair

of robots should come back together. Thus, the following third condition for the tail length L^i must be satisfied:

$$L^i \geq 2\pi\kappa. \tag{4.15}$$

Finally, suppose we have different dragging forces for each robot. How do we handle these by C_1 and C_2? Suppose we have two robots r_i and r_j, where r_i's dragging force is greater than r_j's one. Then, let the total dragging force be double that of r_j's dragging force. Consequentially, a robot r_i reduces the dragging force to tow objects in balance. Similarly, if we have different κ values for two robots, then the sufficient condition is that we take two κ values, where each is proportional to the length of its tail, respectively. Finally, we can compute the minimum length of tails to make a circle with different κ values. Thus Equation (4.15) becomes:

$$L^i \geq 2\pi(\omega_i\kappa_i + \omega_j\kappa_j), \tag{4.16}$$

where ω_i is a tail length ratio for r_i to the overall tail length and ω_j is defined analogously ($\omega_i + \omega_j = 1$).

4.7 Summary of This Section

In this chapter, we considered the problem of collecting multiple objects to satisfy some complex predicate with a coordinated team of mobile robots. Here, we allow a pair of robots to join their tails to act as a single unit. Also we employed individual robots that each of robots tow objects by wrapping their tail around an item. We formulated a general planning problem using logical formulas to deal with complex tasks by such robots. We proved it to be NP-hard, and developed and evaluated practical planning algorithms for the problem. We showed the first known physical demonstration of multiple robots solving manipulation problems in this way.

5. CONCLUSION AND FUTURE WORK

Robots using compliant passive structures (*e.g.,* chains, cords, lines, whips, or lassos) have received little attention in robotics, even though rope-like structures are simple, cheap, and versatile tools. In this dissertation, we investigated how mobile robots can use compliant, unactuated structures for various manipulation tasks. We proposed several methods to deal with the difficulties of modeling and planning. In addition, we solved variants of object manipulation problems wherein multiple classes of objects were transported by cooperative robots using the rope-like structures.

In Section 3, we examined motion primitives for manipulating objects, where the primitives are designed to simplify modeling and planning issues. We demonstrated a diverse range of motion primitives, where each primitive contributes some aspect lacking in the others. We proposed a planning algorithm that seeks a sequence of motion primitives by using a sampling-based motion planning approach, which is coupled with a particle-based representation to treat error propagation of the motions. Our proposed planning algorithm can optimize motion sequences based on specified preferences over a set of objectives, such as execution time, navigation cost, or collision likelihood. We demonstrated how the dynamics of a compliant, unactuated tail can be successfully exploited for manipulation. We showed the effectiveness of an approach that uses simple stochastic models, each associated with a structured primitive. We demonstrated our system with physical robot experiments and the results show that various preferences can be expressed effectively, ultimately being reflected in different mixes of primitives being executed.

In Section 4, we tackled the problem of moving multiple objects to goal locations via a coordinated team of robots. Each robot is equipped with a compliant, unactuated structure attached as a tail; individually, robots can use their tails by wrapping around an object,

securing it by hooking the end of the tail onto itself, and towing the object toward a goal location. In addition, two robots can link the ends of their respective tails to form a conjoined pair that is capable of skimming a region of space and clustering a set of objects together. We devised such motion primitives that can link and unlink robots' tails through a specific series of manoeuvres executed autonomously. Then, we studied the planning problem for efficiently collecting multiple objects and transporting them to goal locations and proposed a general framework using logical formulas to express complex tasks. Lastly, we developed heuristics that give satisfactory solutions in reasonable time. The results we reported included data from physical robots executing plans produced by our planner and collecting objects both by an individual action and by a coupled pair operation.

As a summary, our contributions are fourfold. 1) We made static, quasi-static, and dynamic models for mobile robots using flexible rope-like structures. Then, we showed that dynamic motions are efficient for some objectives. In addition, our proposed algorithm dealt with a diverse set of motions, showing that each motion has a distinct complementary value to the other motions. 2) We studied a coordinated towing system where all robots can be separated or conjoined. Our algorithm dealt with how to form a sub-team and reduced a search space in various environmental settings. 3) We combined multi-robot motion planning with logical specification of plan goals and proposed the first generalized framework using both in a multi-robot manipulation setting. 4) We showed the first known physical demonstration of robots using flexible passive structures to solve manipulation problems: robots use a high-speed motion to manipulate objects; a group of robots changes form operating by a tightly-knit pair or separately.

We expect that our research results will lead to an understanding of how flexible compliant components can be coupled with robots. The proposed research for representing and reasoning the motion planning of a compliant passive tail robot will have a broader impact on several aspects of robotic research: motion planning, minimalist manipulators,

behavior-based control, and multi-robot coordination.

We showed several physical robot experiments in a lab environment. We believe that there are many ways to use these kinds of robots in real world applications.

Consequentially, our proposed techniques can be employed in practical applications: various object manipulation tasks, efficient search and rescue, adaptive environmental monitoring, or localization in complicated environments. We believe that compliant, unactuated tails can enhance the performance of these suggested applications in unknown, unstructured environments. The following sections are ideas for the robotic applications in the near future.

5.1 Suggestions for Future Research: Moving Closer to Practical Applications

In this section, we want to address interesting questions and applications in terms of using compliant unactuated structures for robotic applications.

5.1.1 Cleaning Polluted Water

We know that the oil skimming is useful in preventing the spreading of spilled oil as shown in Figure 5.1(b). Bhattacharya et al. [12] explored how to autonomously control a tethered robot to skim the surface. Similarly, Bhattacharya et al. [30] investigated how to move objects on the surface of water. In this section, we mention a case of a pair of robots manipulating floating objects on the surface water. Water is essential to life, yet water pollution is one of the most serious ecological threats we face today. The United Nations Environment Programme [112] report estimated that there is an average of 46,000 pieces of plastic debris floating near the surface of every square mile of ocean (shown in Figure 5.1(b)). Further instances of the water pollution are in lakes and rivers. For example, Water hyacinth covers lakes and rivers entirely; this blocks sunlight from reaching native aquatic plants, leads lack of oxygen, often killing fish. Thus people use a specialized boat to remove water hyacinth as seen in Figure 5.1(c). Even though people use two boats to

(a) Oil skimming vessels.

(b) Plastic pollution in the oceans.

(c) Two boats are connected by the flexible structure, and they drag a pile of water hyacinth.

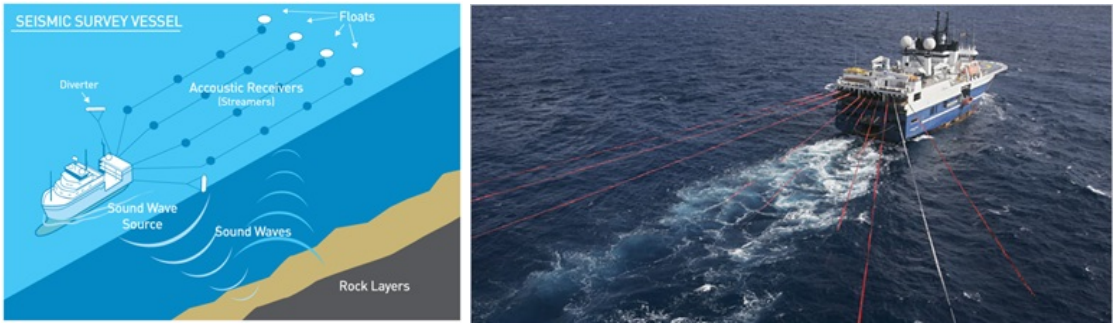
Figure 5.1: Tethered robots can skim oil, drag garbages, and tow water hyacinth.

drag efficiently, considering a large spatial field, individual tail robots might be useful to efficiently remove floating objects such as water hyacinth and plastic debris. In our empirical experiments, the objects on the surface can be surrounded and be moved by the surface vehicle with a tail. This problem can be simplified. Given a large spatial area, many scattered objects, and some number of tail robots, we want to use these tail robots to move all water hyacinth on the surface of a lake to an incineration site. Theoretically, this problem can be solved by our framework in Section 4.

We expect that dragging objects on the water surface using the flexible structures might be a big challenge. There are several ways to manipulate objects directly or indirectly. There are several related works worth examining. First, Leftwich et al. [113] investigated how a passive flexible tail can effect anguilliform swimming. This work might help to understand how tail structures can change the swimming speeds, wake structures, and momentum estimations of such structures. In our empirical studies, dragging objects requires the consideration of many factors such as length, stiffness, or cross sectional areas of structures. Most importantly, since we use various motions, each motion needs different settings; a striking motion needs a high speed, while a hooking motion needs an accurate slow motion. For example, when the robot wants to generate a short turn motion to push objects, the stiffness of the structures are a critical factor affecting the robot speed, momentum and turn radius. Second, Punzmann et al. [114] investigated Quasi-standing Faraday waves to fetch an object at a distance. If we can generate these kinds of waves by using the tail, robots can be used to merely generate motions which will manipulate objects to goal locations.

5.1.2 Environmental Sampling via Flexible Sensor Arrays

Large-scale robotic sampling is a promising application for robotics [115]. Over the past few decades, robotics has emerged as a new tool to collect spatially dense infor-



(a) Seismic survey vessel: mapping out the sea bed rapidly using sound waive sources/receivers [116].

(b) The oil exploration ship Western Neptune pulls streamers, sensors, that stretch four miles behind the ship, and record data from the ocean floor [117].

Figure 5.2: In marine settings, passive compliant components are dragged by the ship for oil exploration. The exploration ship pulls oil sensors that stretch several miles behind the ship.

mation from hazardous environments such as deep oceans [116, 117], lakes [118, 119], seafloors [120], volcanos [121], and hurricanes [122]. The underlying challenge addressed by such systems stems from the fact that the measured data are relatively sparse compared to the large spatial areas/volumes of interest.

Our wish is that robots equipped with a long sensor array can solve environmental monitoring problems efficiently. We believe that one robot dragging a flexible structure as a sensor array gives advantages. First, one robot can gather more information of attributes simultaneously at different locations similar to the exploration ship in Figure 5.2. Second, depending on the robots' motions (*e.g.,* a winding path, a straight path), we can control resolution differently at the specific areas. We think this allows robots to gather efficiently more data at the important area.

5.1.3 Separating Objects

We explored many possible motions using a flexible rope-like structure to manipulate objects in Section 3. Manipulation problems that involve transferring multiple objects to

goal locations arise in many applications and in surprisingly diverse settings. Familiar examples include collecting toys strewn across the floor, removing debris on the surface of a pond, or (more mundanely) handling materials in a warehouse. If the ratio between the number of objects manipulated at a time, and the number of robots required for doing that is small, then a pair of robots can manipulate a large number of objects at one time. However, if we have many different kinds of objects manipulating to different goal locations, we need to separate them first. To separate a large number of objects, Bhattacharya et al. [30] proposed how to separate two kinds of objects using a topological approach, but we are more interested in separating more than two kinds of objects. Our idea is to surround objects using a long flexible rope-like structure, but the robot surrounds objects to separate them by constructing multiple clustering. Then, we can have several snares to represent separated objects. In this case, how do we plan this sequence of snaring motions? Another idea is to use a team of multiple tail robots, whose all tails are physically connected at the end (*cf.* a rat king [123]). Given n classes of objects and m tail robots (assuming $n \leq m$), how do we plan robots to separate n classes of objects?

REFERENCES

[1] The U.S. Defense Advanced Research Projects Agency, "Autonomous Robotic Manipulation — A DARPA Challenge," Mar. 2012, http://www.darpa.mil/program/autonomous-robotic-manipulation.

[2] ——, "DARPA Robotics Challenge," 2015. [Online]. Available: http://www.darpa.mil/program/darpa-robotics-challenge

[3] Wikipedia, "Tool use by animals," 2008. [Online]. Available: https://en.wikipedia.org/wiki/Tool_use_by_animals

[4] N. Correll, K. E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. M. Romano, and P. R. Wurman, "Analysis and observations from the first amazon picking challenge," *IEEE Transactions on Automation Science and Engineering*, vol. PP, no. 99, pp. 1–17, 2016.

[5] YouTube, "Drilling Rig Pipe Connection," 2014, https://youtu.be/KZxUiFFVEAQ.

[6] Wikipedia, "Mine flail," 2005. [Online]. Available: https://en.wikipedia.org/wiki/Mine_flail

[7] Reuters, "Polluted Waters of China," 2015. [Online]. Available: http://www.reuters.com/news/picture/polluted-waters-of-china?articleId=USRTR4WTPW

[8] S. Spielberg, *Indiana Jones*. United States: Lucasfilm Ltd., 1984.

[9] Wikipedia, "Lasso," 2003. [Online]. Available: https://en.wikipedia.org/wiki/Lasso

[10] S. Lee and S. Ditko, *The Amazing Spider-Man*. Marvel Comics, 2003, vol. 2.

[11] B. Donald, L. Gariepy, and D. Rus, "Distributed Manipulation of Multiple Objects using Ropes," in *Proceedings of the International Conference on Robotics and Automation*, San Francisco, CA, USA, Apr. 2000, pp. 450–457.

[12] S. Bhattacharya, H. Heidarsson, G. S. Sukhatme, and V. Kumar, "Cooperative Con-

trol of Autonomous Surface Vehicles for Oil Skimming and Cleanup," in *Proceedings of the International Conference on Robotics and Automation*, Shanghai, China, May 2011, pp. 2374–2379.

[13] S. Thrun and J. J. Leonard, "Robots with Flexible Elements," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer-Verlag Heidelberg, 2008, ch. 13.

[14] R. Deimel and O. Brock, "A Novel Type of Compliant, Underactuated Robotic Hand for Dexterous Grasping," in *Proceedings of Robotics: Science and Systems Conference*, Berkeley, CA, USA, Jun. 2014.

[15] L. U. Odhner, L. P. Jentoft, M. R. Claffee, N. Corson, Y. Tenzer, R. R. Ma, M. Buehler, R. Kohout, R. D. Howe, and A. M. Dollar, "A compliant, underactuated hand for robust manipulation," *The International Journal of Robotics Research*, vol. 33, no. 5, pp. 1–17, 2014.

[16] L. S. Cowan and I. D. Walker, "The Importance of Continuous and Discrete Elements in Continuum Robots," *International Journal of Advanced Robotic Systems*, vol. 10, no. 165, pp. 1–13, 2013.

[17] A. D. Marchese and D. Rus, "Design, kinematics, and control of a soft spatial fluidic elastomer manipulator," *The International Journal of Robotics Research*, vol. 35, no. 7, pp. 840–869, 2016.

[18] G. Smoljkic, G. Borghesan, D. Reynaerts, J. D. Schutter, J. V. Sloten, and E. V. Poorten, "Constraint-Based Interaction Control of Robots Featuring Large Compliance and Deformation," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1252–1260, Oct 2015.

[19] G. Robinson and J. Davies, "Continuum Robots—A State of the Art," in *Proceedings of International Conference on Robotics and Automation*, Detroit, Michigan, USA, May 1999.

[20] S. Chiaverini, G. Oriolo, and I. D. Walker, "Kinematically Redundant Manipulators," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer-Verlag Heidelberg, 2008, ch. 11.

[21] R. J. Webster and B. A. Jones, "Design and Kinematic Modeling of Constant Curvature Continuum," *International Journal of Robotics Research*, vol. 29, no. 13, pp. 1661–1683, 2010.

[22] I. D. Walker, "Robot Strings: Long, Thin Continuum Robots," in *Proceedings of IEEE Aerospace Conference*, Big Sky, MT, USA, Mar. 2013.

[23] M. M. Tonapi, I. S. Godage, A. M. VijayKumar, and I. D. Walker, "Spatial Kinematic Modeling of a Long and Thin Continuum Robotic Cable," in *Proceedings of the International Conference on Robotics and Automation*, Seattle, Washington, USA, May 2015.

[24] I. D. Walker, D. Nahar, S. Verma, M. B. Wooten, and A. D. Kapadia, "Challenges in creating long continuum robots," in *Proceedings of International Conference on Methods and Models in Automation and Robotics (MMAR)*, Miedzyzdroje, Poland, Aug 2016, pp. 339–344.

[25] C. Cohen, B. Hiott, A. D. Kapadia, and I. D. Walker, "Robot tongues in space: continuum surfaces for robotic grasping and manipulation," in *Proceedings of SPIE*, vol. 9836, 2016, pp. 98 362B–98 362B–12.

[26] K. M. Lynch and M. T. Mason, "Stable Pushing: Mechanics, Controllability, and Planning," *The International Journal of Robotics Research*, vol. 15, no. 6, pp. 533–556, Dec. 1996.

[27] T. Meriçli, M. Veloso, and H. L. Akin, "Push-manipulation of complex passive mobile objects using experimentally acquired motion models," *Autonomous Robot*, vol. 38, pp. 317–329, 2015.

[28] J. Fink, M. A. Hsieh, and V. Kumar, "Multi-Robot Manipulation via Caging in

Envirionments with Obstacles," in *Proceedings of the International Conference on Robotics and Automation*, Pasadena, CA, USA, May 2008, pp. 1471–1476.

[29] W. H. Huang, E. P. Krotkov, and M. T. Mason, "Impulsive Manipulation," in *Proceedings of International Conference on Robotics and Automation*, May 1995.

[30] S. Bhattacharya, S. Kim, H. Heidarsson, G. S. Sukhatme, and V. Kumar, "A topological approach to using cables to separate and manipulate sets of objects," *The International Journal of Robotics Research*, vol. 34, no. 6, pp. 799–815, 2015.

[31] D. Rus, B. Donald, and J. Jennings, "Moving furniture with teams of autonomous robots," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, Aug 1995, pp. 235–242.

[32] J. Fink, N. Michael, and V. Kumar, "Composition of Vector Fields for Multi-Robot Manipulation via Caging," in *Proceedings of Robotics: Science and Systems Conference*, Atlanta, Georgia, USA, Jul. 2007.

[33] P. Cheng, J. Fink, and V. Kumar, "Abstractions and Algorithms for Cooperative Multiple Robot Planar Manipulation," in *Proceedings of Robotics: Science and Systems Conference*, Seattle, USA, Jun. 2009, pp. 143–150.

[34] G. Sartoretti, S. Shaw, and M. A. Hsieh, "Distributed planar manipulation in fluidic environments," in *Proceedings of International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, May 2016, pp. 5322–5327.

[35] C. Kube and E. Bonabeau, "Cooperative transport by ants and robots," *Robotics and Autonomous Systems*, vol. 30, no. 1–2, pp. 85 – 101, 2000.

[36] S. Wilson, T. P. Pavlic, G. P. Kumar, A. Buffin, S. C. Pratt, and S. Berman, "Design of ant-inspired stochastic control policies for collective transport by robotic swarms," *Swarm Intelligence*, vol. 8, no. 4, pp. 303–327, 2014.

[37] T. Maneewarn and P. Detudom, "Mechanics of cooperative nonprehensile pulling by multiple robots," in *Proceedings of IEEE/RSJ International Conference on In-*

telligent Robots and Systems (IROS), Alberta, Canada, Aug. 2005, pp. 2004–2009.

[38] P. Cheng, J. Fink, V. Kumar, and J.-S. Pang, "Cooperative Towing With Multiple Robots," *Journal of Mechanisms and Robotics*, vol. 1, no. 1, 2008.

[39] Q. Jiang and V. Kumar, "The inverse kinematics of 3-D towing," *Advances in Robot Kinematics: Motion in Man and Machine*, pp. 321–328, 2010.

[40] N. Michael, J. Fink, and V. Kumar, "Cooperative manipulation and transportation with aerial robots," *Autonomous Robots*, vol. 30, no. 1, pp. 73–86, 2011.

[41] J. Fink, N. Michael, S. Kim, and V. Kumar, "Planning and control for cooperative manipulation and transportation with aerial robots," *The International Journal of Robotics Research*, vol. 30, no. 3, pp. 321–328, 2011.

[42] M. Levihn, T. Igarashi, and M. Stilman, "Multi-robot multi-object rearrangement in assignment space," in *Proceedings of IEEE Conference on Intelligent Robots and Systems*, Vilamoura, Algarve, Portugal, Oct. 2012, pp. 5255–5261.

[43] N. Fujii, R. Inoue, Y. Takebe, and J. Ota, "Multiple robot rearrangement planning using a territorial approach and an extended project scheduling problem solver," *Advanced Robotics*, vol. 24, no. 1-2, pp. 103–122, 2010.

[44] N. Oyama, Z. Liu, L. B. Gueta, and J. Ota, "Rearrangement task of multiple robots using task assignment applicable to different environments," in *Proceedings of International Conference on Robotics and Biomimetics (ROBIO)*, Tianjin, China, Dec. 2010, pp. 300–305.

[45] R. Inoue, N. Fujii, R. Takano, and J. Ota, "Realization of a multiple object rearrangement task with two multi-task functional robots," *Advanced Robotics*, vol. 25, no. 11-12, pp. 1365–1383, 2011.

[46] A. Yamashita, J. Sasaki, J. Ota, and T. Arai, "Cooperative manipulation of objects by multiple mobile robots with tools," in *Proceedings of the 4th Japan-France/2nd Asia-Europe Congress on Mechatronics*, Fukuoka, Japan, 1998, pp. 310–315.

[47] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management Science*, vol. 6, no. 1, pp. 80–91, 1959.

[48] T. K. Ralphs, L. Kopman, W. R. Pulleyblank, and L. E. Trotter, "On the capacitated vehicle routing problem," *Mathematical Programming*, vol. 94, no. 2, pp. 343–359, 2003.

[49] N. Mathew, S. L. Smith, and S. L. Waslander, "Planning Paths for Package Delivery in Heterogeneous Multirobot Teams," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 4, pp. 1298–1308, Oct 2015.

[50] B. Coltin and M. Veloso, "Optimizing for transfers in a multi-vehicle collection and delivery problem," in *Proceedings of Distributed Autonomous Robotic Systems*, Baltimore, Maryland, 2012, pp. 91–103.

[51] R. Luna and K. E. Bekris, "Efficient and complete centralized multi-robot path planning," in *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, California, USA, Sept 2011, pp. 3268–3275.

[52] K. Solovey and D. Halperin, "k-color multi-robot motion planning," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 82–97, 2014.

[53] J. van den Berg, J. Snoeyink, M. Lin, and D. Manocha, "Centralized Path Planning for Multiple Robots: Optimal Decoupling into Sequential Plans," in *Proceedings of Robotics: Science and Systems*, Seattle, USA, 2009.

[54] M. Turpin, K. Mohta, N. Michael, and V. Kumar, "Goal assignment and trajectory planning for large teams of interchangeable robots," *Autonomous Robots*, vol. 37, no. 4, pp. 401–415, 2014.

[55] G. Wagner, M. Kang, and H. Choset, "Probabilistic path planning for multiple robots with subdimensional expansion," in *Proceedings of International Conference on Robotics and Automation (ICRA)*, Saint Paul, Minnesota, USA, May 2012, pp. 2886–2892.

[56] G. Wagner and H. Choset, "Subdimensional expansion for multirobot path planning," *Artificial Intelligence*, vol. 219, pp. 1–24, 2015.

[57] N. Mathew, S. L. Smith, and S. L. Waslander, "Multirobot Rendezvous Planning for Recharging in Persistent Tasks," *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 128–142, 2015.

[58] S.-G. Roh, J. H. Park, Y. K. Song, K. Yang, M. Choi, H.-S. Kim, H. Lee, and H. R. Choi, "Flexible Docking Mechanism Using Combination of Magnetic Force with Error-Compensation Capability," in *Proceedings of IEEE Conference on Automation Science and Engineering*, Washington DC, USA, Aug. 2008.

[59] Y.-H. Kim, S.-W. Lee, H. S. Yang, and D. A. Shell, "Toward autonomous robotic containment booms: visual servoing for robust inter-vehicle docking of surface vehicles," *Intelligent Service Robotics*, vol. 5, no. 1, pp. 1–18, 2012.

[60] T. Libby, T. Y. Moore, E. Chang-Siu, D. Li, D. J. Cohen, A. Jusufi, and R. J. Full, "Tail-assisted pitch control in lizards, robots and dinosaurs," *Nature Letter*, vol. 481, pp. 181–186, 2012.

[61] E. Chang-Siu, T. Libby, M. Tomizuka, and R. J. Full, "A lizard-Inspired Active Tail Enables Rapid Maneuvers and Dynamic Stabilization in a Terretrial Robot," in *Proceedings of the International Conference on Intelligent Robots and Systems*, San Fransisco, CA, USA, Sep. 2011.

[62] J. Zhao, T. Zhao, N. Xi, F. J. Cintron, M. W. Mutka, and L. Xiao, "Controlling Aerial Maneuvering of a Miniature Jumping Robot Using Its Tail," in *Proceedings of the International Conference on Intelligent Robots and Systems*, Tokyo, Japan, Nov. 2013.

[63] W. S. Rone and P. Ben-Tzvi, "Continuum Robotic Tail Loading Analysis for Mobile Robot Stabilization and Maneuvering," in *Proceedings of International Design Engineering Technical Conferences & Computers and Information in Engineering*

Conference, Buffalo, New York, USA, Aug. 2014.

[64] J. Ackerman, X. Da, and J. Seipel, "Mobility of Legged Robot Locomotion with Elastically-suspended Loads over Rough Terrain," in *Proceedings of the Fifteenth International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines*, Baltimore, MD, USA, Jul. 2012.

[65] A. Patel and M. Brrae, "Rapid Turning at High-Speed: Inspirations from the Cheetah's Tail," in *Proceedings of the International Conference on Intelligent Robots and Systems*, Tokyo, Japan, Nov. 2013.

[66] N. J. Kohut, A. O. Pullin, D. W. Haldane, D. Zarrouk, and R. S. Fearing, "Precise Dynamic Turning of a 10 cm Legged Robot on a Low Friction Surface Using a Tail," in *Proceedings of the International Conference on Robotics and Automation*, Karlsruhe, Germany, May 2013.

[67] R. Briggs, J. Lee, M. Haberland, and S. Kim, "Tails in Biomimetic Design: Analysis, Simulation, and Experiment," in *Proceedings of the International Conference on Intelligent Robots and Systems*, Vilamoura, Algarve, Portugal, Oct. 2012.

[68] G.-H. Liu, H.-Y. Lin, H.-Y. Lin, S.-T. Chen, and P.-C. Lin, "A Bio-Inspired Hopping Kangaroo Robot with An Active Tail," *Journal of Bionic Engineering*, vol. 11, no. 4, pp. 541–555, Oct. 2014.

[69] T. Libby, A. M. Johnson, E. Chang-Siu, R. J. Full, and D. E. Koditschek, "Comparative Design, Scaling, and Control of Appendages for Inertial Reorientation," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1380–1398, Dec 2016.

[70] Y.-H. Kim and D. A. Shell, "Using a compliant, unactuated tail to manipulate objects," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 223–230, Jan. 2017.

[71] D. A. Shell and M. J. Matarić, "Behavior-Based Methods for Modeling and Structuring Control of Social Robots," in *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation*, R. Sun, Ed. Cambridge University Press,

2005, ch. 11.

[72] O. C. Jenkins and M. J. Matarić, "Performance-Derived Behavior Vocabularies: Data-driven Acqusition of Skills from Motion," *International Journal of Humanoid Robotics*, vol. 1, no. 237, May 2004.

[73] K. Hauser, T. Bretl, K. Harada, and J.-C. Latombe, "Using Motion Primitives in Probabilistic Sample-Based Planning for Humanoid Robots," in *Proceedings of Workshop on the Algorithmic Foundations of Robotics*, New York City, USA, 2006.

[74] M. J. Powell, H. Zhao, and A. D. Ames, "Motion Primitives for Human-Inspired Bipedal Robotic Locomotion: Walking and Stair Climbing," in *Proceedings of IEEE Conference on Robotics and Automation (ICRA)*, Saint Paul, Minnesota, USA, 2012.

[75] S. M. LaValle and J. J. Kuffner, "Randomized Kinodynamic Planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[76] M. Pivtoraiko and A. Kelly, "Efficient Constrained Path Planning Via Search In State Lattices," in *Proceedings of International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, Munich, Germany, Sep. 2005.

[77] J. Butzke, K. Sapkota, K. Prasad, B. MacAllister, and M. Likhachev, "State Lattice with Controllers: Augmenting Lattice-Based Path Planning with Controller-Based Motion Primitives," in *Proceedings of International Conference on Intelligent Robots and Systems*, Chicago, USA, Sep. 2014.

[78] V. Vonásek, L. winkler, J. Liedke, M. Saska, karel Kosnar, and L. Preucil, "Fast on-board motion planning for modular robots," in *Proceedings of the International Conference on Robotics and Automation*, Hong Kong, China, May 2014.

[79] A. Krontiris and K. Bekris, "Dealing with difficult instances of object rearrangement," in *Proceedings of Robotics: Science and Systems*, Rome, Italy, July 2015.

[80] A. Gray, Y. Gao, T. Lin, J. K. Hedrick, H. E. Tseng, and F. Borrelli, "Predictive Con-

trol for Agile Semi-Autonomous Ground Vehicles using Motion Primitives," in *Proceedings of American Control Conference*, Fairmont Queen Elizabeth, Montréal, Canada, Jun. 2012.

[81] A. A. Paranjape, K. C. Meier, X. Shi, S.-J. Chung, and S. Hutchinson, "Motion primitives and 3D path planning for fast flight through a forest," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 357–377, 2015.

[82] M. Gupta, J. Ḿuller, and G. S. Sukhatme, "Using Manipulation Primitives for Object Sorting in Cluttered Environments," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 608–614, Apr. 2015.

[83] M. R. Dogar and S. S. Srinivasa, "A Planning Framework for Non-Prehensile Manipulation under Clutter and Uncertainty," *Autonomous Robots*, vol. 33, no. 3, pp. 217–236, Jun. 2012.

[84] M. Phillips, B. Cohen, S. Chitta, and M. Likhachev, "E-graphs: Bootstrapping planning with experience graphs," in *Proceedings of Robotics: Science and Systems*, Sydney, Australia, Jul. 2012.

[85] S. M. LaValle, *Planning Algorithms*. New York, NY, USA: Cambridge University Press, 2006.

[86] D. Berenson, P. Abbeel, and K. Goldberg, "A robot path planning framework that learns from experience," in *Proceedings of International Conference on Robotics and Automation*, Saint Paul, Minnesota, USA, May 2012.

[87] A. Bry and N. Roy, "Rapidly-exploring Random Belief Trees for Motion Planning Under Uncertainty," in *Proceedings of the International Conference on Robotics and Automation*, Shanghai, China, May 2011.

[88] C. Barrett and C. Tinelli, "CVC3," in *Proceedings of the* 19^{th} *International Conference on Computer Aided Verification (CAV '07)*, ser. Lecture Notes in Computer Science, W. Damm and H. Hermanns, Eds., vol. 4590. Berlin, Germany: Springer-

Verlag, Jul. 2007, pp. 298–302.

[89] L. De Moura and N. Bjørner, "Z3: An efficient SMT solver," in *Proceedings of International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, Budapest, Hungary, Mar. 2008, pp. 337–340.

[90] T. Bouton, D. Caminha B. de Oliveira, D. Déharbe, and P. Fontaine, "veriT: An Open, Trustable and Efficient SMT-Solver," in *Automated Deduction – CADE-22: 22nd International Conference on Automated Deduction, Montreal, Canada, August 2-7, 2009. Proceedings*, R. A. Schmidt, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 151–156.

[91] G. Fedyukovich, O. Sery, and N. Sharygina, "eVolCheck: Incremental Upgrade Checker for C," in *Proceedings of International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, Rome, Italy, Mar. 2013, pp. 292–307.

[92] L. De Moura and N. Bjørner, "Satisfiability Modulo Theories: Introduction and Applications," *Commun. ACM*, vol. 54, no. 9, pp. 69–77, Sep. 2011.

[93] W. N. N. Hung, X. Song, J. Tan, X. Li, J. Zhang, R. Wang, and P. Gao, "Motion planning with Satisfiability Modulo Theories," in *Proceedings of International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 113–118.

[94] S. Nedunuri, S. Prabhu, M. Moll, S. Chaudhuri, and L. E. Kavraki, "SMT-based synthesis of integrated task and motion plans from plan outlines," in *Proceedings of International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 655–662.

[95] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "Incremental Task and Motion Planning: A Constraint-Based Approach," in *Proceedings of Robotics: Science and Systems Conference*, Michigan, USA, Jun. 2016.

[96] I. Saha, R. Ramaithitima, V. Kumar, and G. J. P. andSanjit A. Seshia, "Automated

composition of motion primitives for multi-robot systems from safe LTL specifications," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2014, pp. 1525–1532.

[97] T. Igarashi and M. Stilman, "Homotopic Path Planning on Manifolds for Cabled Mobile Robots," in *Proceedings of International Workshop on the Algorithmic Foundations of Robotics*, Singapore, Dec. 2010.

[98] J.-H. Kim and D. A. Shell, "A new model for self-organized robotic clustering: Understanding boundary induced densities and cluster compactness," in *Proceedings of International Conference on Robotics and Automation (ICRA)*, Seattle, WA, USA, May 2015, pp. 5858–5863.

[99] J. Wu, I. Yildirim, J. J. Lim, W. T. Freeman, and J. B. Tenenbaum, "Galileo: Perceiving Physical Object Properties by Integrating a Physics Engine with Deep Learning," in *Proceedings of International Conference on Neural Information Processing Systems (NIPS)*, Cambridge, MA, USA, 2015, pp. 127–135.

[100] S. Mahadevan and J. Connell, "Automatic Programming of Behavior-based Robots using Reinforcement Learning," in *Proceedings of AAAI Conference on Artificial Intelligence*, Anaheim, California, USA, Jul. 1991.

[101] J. Morimoto and K. Doya, "Reinforcement learning of dynamic motion sequence: Learning to stand up," in *Proceedings of IEEE Conference on Intelligent Robots and Systems*, Victoria, BC, Oct. 1998.

[102] V. Soni and S. Singh, "Reinforcement learning of hierarchical skills on the Sony AIBO robot," in *Proceedings of Conference on Development and Learning (ICDL)*, May 2013.

[103] C. Daniel, G. Neumann, and J. Peters, "Learning Concurrent Motor Skills in Versatile Solution Spaces," in *Proceedings of IEEE Conference on Intelligent Robots and Systems*, Vilamoura, Algarve, Portugal, Oct. 2012.

[104] J. Z. Kolter and A. Y. Ng, "Policy Search via the Signed Derivative," in *Proceedings of Robotics: Science and Systems Conference*, Seattle, USA, Jun. 2009.

[105] J. Kober, E. Oztop, and J. Peters, "Reinforcement Learning to Adjust Robot Movements to New Situations," in *Proceedings of Robotics: Science and Systems Conference*, Zaragoza, Spain, Jun. 2010.

[106] J. Michels, A. Saxena, and A. Y. Ng, "High Speed Obstacle Avoidance using Monocular Vision and Reinforcement Learning," in *Proceedings of IEEE Conference on Machine Learning*, Bonn, Germany, 2005.

[107] P. R. Giordano and M. Vendittelli, "Shortest Paths to Obstacles for a Polygonal Dubins Car," *IEEE Transactions on Robotics*, vol. 25, no. 5, pp. 1184–1191, 2009.

[108] T. Meriçli, M. Veloso, and H. L. Akin, "Achievable Push-Manipulation for Complex Passive Mobile Objects using Past Experience," in *Proceedings of International Conference on Autonomous Agents and Multiagent Systems*, Saint Paul, Minnesota, USA, May 2013.

[109] J. K. Lenstra and A. H. G. R. Kan, "Complexity of vehicle routing and scheduling problems," *Networks*, vol. 11, no. 2, pp. 221–227, Jun. 1981.

[110] H. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistic Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[111] R. H. Teshnizi and D. A. Shell, "Planning motions for a planar robot attached to a stiff tether," in *Proceedings of International Conference on Robotics and Automation*, Stockholm, Sweden, May 2016, pp. 2759–2766.

[112] The United Nations Environment Programme, "Ecosystems and Biodiversity in Deep Waters and High Seas," United Nations Environment Programme, Tech. Rep. 178, 2006.

[113] M. C. Leftwich, E. D. Tytell, A. H. Cohen, and A. J. Smits, "Wake structures behind a swimming robotic lamprey with a passively flexible tail," *Journal of Experimental*

Biology, vol. 215, no. 3, pp. 416–425, 2012.

[114] H. Punzmann, N. Francois, H. Xia, G. Falkovich, and M. Shats, "Generation and reversal of surface flows by propagating waves," *Nature Physics*, vol. 10, pp. 658–663, Jun. 2014.

[115] V. Kumar, D. Rus, and G. S. Sukhatme, "Networked Robots," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer-Verlag Heidelberg, 2008, ch. 41.

[116] Popular Mechanics, "This Colossal Oil-Hunter Is Largest Mobile Manmade Object in the World," 2016. [Online]. Available: http://www.popularmechanics.com/technology/infrastructure/a19081/polarcus-largest-manmade-mobile-object/

[117] The New York Times, "Drilling Deep in the Gulf of Mexico," 2006. [Online]. Available: http://www.nytimes.com/2006/11/08/business/worldbusiness/08gulf.html

[118] Y.-H. Kim, D. A. Shell, C. Ho, and S. Saripalli, "Spatial Interpolation for Robotic Sampling: Uncertainty with two Models of Variance," in *Proceedings of International Symposium on Experimental Robotics*, Quebec, Canada, Jun. 2012.

[119] Y.-H. Kim and D. A. Shell, "Distributed Robotic Sampling of Non-Homogeneous Spatio-Temporal Fields via Recursive Geometric Sub-division," in *Proceedings of the International Conference on Robotics and Automation*, Hong Kong, China, May 2014.

[120] L. Whitcomb, "Underwater robotics: Out of the research laboratory and into the field," in *Proceedings of the International Conference on Robotics and Automation*, San Francisco, CA, USA, May 2000.

[121] S. Guccione, G. Muscato, G. Nunnari, G. Virk, A. Azad, A. Semerano, T. White, and C. Glazebrook, "Robots for volcanos: The state of the art," in *Proceedings of the International Conference on Climbing and Walking Robots (CLAWAR)*, Oct. 2000.

[122] P.-H. Lin and C.-S. Lee, "The Eyewall-Penetration Reconnaissance Observation of Typhoon Longwang (2005) with unmanned aerial vehicle, aerosonde," *J. Atmos. Oceanic Technol*, vol. 25, no. 1, pp. 15–25, 2008.

[123] Wikipedia, "Rat king," 2009. [Online]. Available: https://en.wikipedia.org/wiki/Rat_king